TURING

The Golden Ticket P NP and the Search for the Impossible

寻找金券、旅行推销员问题
地图着色问题、数独游戏、团问题

《出版人周刊》、《科学》、《纽约客》、《新科学人》热评

互联网之父Vint Cerf力荐

可能与不可能的边界

P/NP问题趣史

[美] Lance Fortnow◎著

杨帆◎译

人民邮电出版社
POSTS & TELECOM PRESS

版权信息

书名：可能与不可能的边界：P/NP问题趣史

作者：Lance Fortnow

译者：杨帆

ISBN：978-7-115-33566-1

目录

版权声明

Original edition, entitled The Golden Ticket: P, NP, and the Search for the Impossible by Lance Fortnow, ISBN: 978-0-691-15649-1 , published by Princeton University Press.

献词

献给Marcy、Annie 和Molly，愿他们知道我是做什么的，以及为何而做。

英国著名科学家臭臭教授发明了一个机器，不用打开糖果包装，它就能立刻告诉你里面有没有金券。机器的机械手臂出手如电，一抓一个准儿，不会漏掉哪怕藏有一点点金子的任何东西。目前看来，它解决了所有的问题。

——罗尔德·达尔，《查理和巧克力工厂》

前言

近半数的美国人都拥有智能手机。智能手机也是计算机，其计算能力比几十年前的超级计算机还要强。计算机将世界上的信息呈现在我们眼前，也帮我们梳理信息。计算机让人们可以彼此交流，无论什么身份，地处何方。计算机能执行数量巨大的运算，从模拟宇宙事件到调度复杂的航线。计算机可以识别人的声音、面孔和动作。计算机可以获悉人们的喜好，并据此推荐图书、音乐和电影。在不远的将来，借助计算机技术，无人驾驶的汽车将随处可见。这么说，计算机简直无所不能。

真是这样吗？在这本书里，我们将探讨许多计算问题，其中一部分可能永远都无法用简单的计算得到答案。试着解答它们是计算机科学，乃至整个数学和科学领域最重要的挑战。人们给这些问题起了一个有些奇怪的名字：P/NP问题。

P/NP是克雷数学研究所公布的7个千禧年数学难题之一，该研究所为求解这道难题设立了百万美元的奖金。不过，P/NP问题的意义远不止于此。

P指的是用计算机能很快求解的问题，NP指的是我们想找到最优解的问题。如果P=NP，那么我们将很容易找到任意给定问题的解。P=NP意味着我们所了解的社会将发生巨变，医学、科学、娱乐和人类社会一切任务的自动化程度都将立即发生质的飞跃。

相反，如果P≠NP，那么总会有部分问题无法迅速地被解决。那也没有关系，因为我们可以根据具体情况研发出某些技术来解决这些问题。P≠NP意味着不可能用自动化的方法解决所有问题。然而，知道哪些工具不好用也有助于人们找到更好用的工具。

2008年8月，《ACM通讯》的主编莫舍·瓦迪约我写一篇关于P/NP问题的文章。ACM（美国计算机协会）是一个为计算机研究学者和从业人员服务的重要社团，《ACM通讯》则是该协会刊登文章的主要杂志。

一开始我想把写稿的事推给另一位计算机科学家，但后来我还是答应了。当时莫舍是这么劝说我的：“如果那帮物理学家可以写关于弦理论的畅销文章（和图书），那我们也可以向公众解释计算复杂度理论目前

的进展，我希望如此。”于是我写了一篇文章，该文章以《ACM通讯》的读者为主要受众，不仅介绍了P/NP问题的现状（基本可以概括为“悬而未决”），也讲了一些人们在处理困难问题时积累的技巧。“P/NP问题的现状”（The Status of the P versus NP Problem）发表在2009年9月的《ACM通讯》上，它很快就成为该刊物创刊以来下载次数最多的文章。

关于P/NP问题，我觉得还有很多故事可讲，而那篇文章的大受欢迎，似乎表明是时候面向更广的受众（而不仅是科学家们）来讲述这些故事了。

我将那篇短文作为本书的框架结构，将原来文章的各个部分扩展为现在的章节。我还受到了史蒂芬·霍金的《时间简史》的启发：该书尽量绕开晦涩的公式和术语，采用生动的例子和故事来解释物理。我试图以同样的方式来讲解P/NP问题，借此探讨P/NP问题的本质和重要意义。

本书没有给出P/NP问题的正式定义，有很多教科书和网站都详细论述了P和NP的定义及技术结论。本书旨在让你对计算科学的潜能和局限性有更多的了解，这非常有好处，毕竟计算机如今已成为人类生活不可或缺的部分了。

向P和NP进发吧！

兰斯·福特诺

于伊利诺伊州埃文斯顿

致谢

首先感谢Moshe Vardi，是他鼓励我写作，他也是我发表在《ACM通讯》杂志上的“P/NP问题的现状”一文的编辑。这篇文章受欢迎的程度让我萌发了把它扩展成一本科普书的念头。

跟我一块写博客的Bill Gasarch不断鼓励我，并仔细审阅了全书各章的初稿。Alana Lidawer和John、Jim，以及Chris Purtilo也阅读了全书的初稿，并提出了许多宝贵意见。KuanLing Chen、Josh Grochow、Ralph Hansen、Adam Kalinich、David Pennock和Rahul Santhanam分别对本书的部分章节提出了许多宝贵的意见。

Manuel Blum、Steve Cook、David Johnson、Leonid Levin和Albert Meyer分别以其个人独到的视角向我讲述了P/NP问题的早期历史。Alexander Razborov对俄罗斯历史的介绍对我帮助很大。

这本书的写作离不开我的生活圈子。作为计算机科学家，我在工作和生活中会与其他研究人员、学生，以及数不清的人交流，这种交流也让我受益匪浅。在此特别感谢加州大学伯克利分校、麻省理工学院、芝加哥大学、阿姆斯特丹的数学与计算机科学中心、NEC研究院、丰田工业大学芝加哥分校以及西北大学的同行们，与他们真挚而友好的讨论让我获益良多。

另外，我还要特别感谢两个人，他们对我早年P/NP问题观念的形成影响很大：Juris Hartmanis，我在康奈尔大学读本科期间首次从他这里接触到了P和NP，还有Michael Sipser，他是我在加州大学伯克利分校和麻省理工学院的博士学位指导老师和好朋友。

在为第6章的地图填色问题寻找例子时，我在网上寻求了帮助，感谢那些做出回应的人：Chris Bogart、HsienChih Chang、Pálvölgyi Dömötör、David Eppstein、Lukasz Grabowski、Gil Kalai、Charles Martel以及Derrick Stolee。

写作期间，我在西北大学的Robert R. McCormick工程与应用科学学院担任电子工程和计算机科学教授。西北大学十分鼓励向公众传播知识的著书活动。我充分利用了教职的特权，尤其是利用了学校图书馆丰富的资

源，包括纸质文献和电子文献。西北大学的教职工是最棒的，我的行政助理Marjorie Reyes对我帮助同样很大。

普林斯顿大学出版社的编辑Vickie Kearn为我提供了悉心的指导，并在著书的各个阶段仔细审阅了手稿，让这本书变得更好。我还想感谢Vickie的助手Quinn Fusting，以及出版社的全体工作人员。

最大的感谢留给我的家人：妻子Marcy、两个女儿Annie和Molly，感谢她们对我的爱与鼓励。

第1章 金券

一个糖果厂老板决定推出一个活动，将五张金券藏到巧克力的包装里，而这种巧克力每年的产量数以千万计。找到金券的人将得到一次珍贵的参观工厂的机会。

如何找到这些金券？你可以买尽可能多的巧克力。你可能会试试用磁铁，可惜金没有磁性。或者你可以雇用数千人，让他们每人筛查一小堆巧克力。这听起来很傻，但是小姑娘维露卡·索尔特就要这么做，因为她特别想得到一张金券，去参观威利·旺卡的巧克力工厂。

维露卡的父亲索尔特先生是个富商，他决定买光他能找到的巧克力。这还不够，就算有堆积如山的巧克力，要从中找到小小的金券也很困难。索尔特先生也有一家工厂，他不惜动用工厂的工人，终于找到了一张金券。他对记者讲述了找到金券的过程：

> 我是做花生生意的，知道吧，我有大约100个女工为我剥花生，然后将它们做成烤花生米和腌花生米。她们整天就坐在那儿剥花生。所以我跟她们说："好了姑娘们，不要剥花生了，大家开始给我剥这些破糖纸吧！"然后她们就剥。我让工厂的每一个工人都铆足了劲地撕掉巧克力的包装纸，从早干到晚。
>
> 但是三天过去了，我们还是没走运。哦，那可真够呛！我可怜的小维露卡越来越暴躁，每次我一回家她就朝我嚷嚷："我的金券在哪儿？我要我的金券！"她撒泼又打滚儿，踢腿又叫喊，实在招人烦。我可不希望看到我的小宝贝这么不高兴，所以我决定一直找，不找到她要的东西誓不罢休。终于，在第四天的晚上，一个女工大叫："我找到金券了！"然后我说："把它给我，快！"她给了我，然后我跑着回家把它交给了亲爱的维露卡，她高兴得合不拢嘴。我们家又变得其乐融融了。

和索尔特先生一样，无论你打算怎么找那张金券，你都需要大量时间、金钱，或者运气。也许有一天，有人能发明出一个快速找到金券的便宜装置，也许这样的装置并不存在。

然而，1000万对于今天的计算机来说只是很小的数字。如果你把糖果数

字化，录入一个数据库，现在的电脑只用不到一秒就能把它找一遍。虽然计算机比人快得多，但它面对的问题的规模也比在糖果里找金券大得多。

现在最大的电子数据集合规模有多大？比如，整个互联网，考虑到所有视频、音频、电子邮件及其他一切，总的信息量差不多有1 000 000 000 000 000 000字节，最多相差几个0。一个字节大致对应键盘上的一个字符。这个数很大，但记住，计算机也很快。一般的笔记本电脑每秒可以执行1万亿次操作，这样算来，理论上只需要不到4个月就能搜完整个互联网的内容，前提是你能把整个互联网装到你电脑的内存里。Google每时每刻都在搜索整个互联网，它使用了几十万台快速的计算机。

如果计算机可以很快地搜遍整个互联网，看起来好像我们就解决了这个找金券问题的电子版。但是，计算机不仅要帮人们搜索已有的数据，还要搜索问题的所有可能解。

认识一下可怜的旅行推销员Mary，她来自华盛顿特区，为美国木槌集团公司工作。她需要从家乡旅行到48个州的首府，向各州法院推销木槌。木槌公司为了削减成本，让Mary找到通过所有城市的最短路径。Mary画了一张图，写写画画了一会儿，制订了一个不错的路线。



图**1-1**　旅行推销员问题的地图

1 1英里约合1.609千米。——编者注

差旅部门的人想让Mary试试能否找到另一条路线，把路程缩短到11 000英里以下。Mary写了个计算机程序，试图穷举所有可能的路线，找出最短的一条，但是一周以后，程序还没跑完。Mary坐下来开始算数。作为第一站的城市有48种选择，然后从剩下的47个城市中选一个作为第二站，再从剩下的46个城市中选一个，以此类推。可能的路径共有48×47×46×…×2×1种，也就是下面这个62位数：

12 413 915 592 536 072 670 862 289 047 373 375 038 521 486 354 677 760 000 000 000

即使计算机计算一条路线的时间等于光通过最小的原子直径的时间（大约0.000 000 000 000 000 000 33秒），仍然需要十亿亿亿倍于宇宙年龄的时间才能算完。难怪Mary的电脑算了一周还没有完。Mary想知道有没有比穷举更好的方法找到最佳路线，就像在所有可能行程的“巧克力山”里面刨出那张小小的金券。

这就是本书最基本的问题：P/NP问题。其中的一个实例就是能否为旅行推销员找到最短路径。P和NP自有其十分专业的定义，但是把它们看做概念比看做数学对象更好。NP是存在解的问题的集合，P则是能很快找到解的问题的集合。P=NP意味着我们能总是很快地计算出任何问题的解，当然也包括找到旅行推销员的最短路径。相反，P≠NP意味着我们不能。

1.1　划分的难题

看下边38个数字：

14 175, 15 055, 16 616, 17 495, 18 072, 19 390, 19 731, 22 161, 23 320, 23 717, 26 343, 28 725, 29 127, 32 257, 40 020, 41 867, 43 155, 46 298, 56 734, 57 176, 58 306, 61 848, 65 825, 66 042, 68 634, 69 189, 72 936, 74 287, 74 537, 81 942, 82 027, 82 623, 82 802, 82 988, 90 467, 97 042, 97 507, 99 564

这38个数字之和为2 000 000。你能把它们平分成两组，每组19个数字之和分别为1 000 000吗？你可以使用计算器、电子表格或写一个计算机程序。（答案在本章最后。）

不那么简单，是吧？把这些数分成两组有170亿种方式。如果程序编得巧妙，使用当今较快的计算机能够找到一个解。但如果给你3800个数，或者3800万个数呢？短小的计算机程序可没法给出答案了！

这只是个无意义的数学谜题吗？就算存在一个厉害的计算机程序，它能解决这个平分数组的问题（假设有解），那又如何呢？如果是这样的话，我们能用这个程序做更多的事。这个程序能解决所有的问题，包括旅行推销员问题。这个简短的难题抓住了P/NP问题的本质：一个程序如果能解决这个难题的复杂版本，那么它也能解出任意问题。

1.2 手

你的手是最不可思议的工程装置，它能戳、抓和指，能系鞋带，能射箭，还能弹钢琴、拉小提琴，能变戏法，能驾驶车、船、火车或飞机。你的手可以握住其他人的手，或跟他们玩拇指相扑。手可以比划出信号语言，也能通过写字或打字来交流。手可以轻抚，也能重击。手可以使用修理钟表的精密工具，也能操作链条锯。有才华的人的双手可以创造艺术杰作，写出音乐或诗歌。人类取得的几乎所有成就，都离不开双手。

一只手有27块骨头，5根手指，包括最重要的拇指。手具有结构复杂的神经、肌腱和肌肉，这些都包裹在富有弹性的皮肤里。然而，这一不可思议的装置，自然造物的杰作，却不能自己做事，而只能执行人脑的指令。死人的手平平无奇，做不了任何事情。

手就是自然的硬件，硬件本身不能做什么。手需要软件（也就是大脑指令）来控制，软件告诉它如何执行和实现大脑希望它做的事情。

松冈容子是华盛顿大学的机器人学教授，她带领科研小组制作了一个解剖学上正确无误的机械手，其手指有和人类手指同样的动作自由度。理论上她的机械手可以完成人手能做的任何事情，但实际上，它只能完成很简单的任务，因为写一个计算机程序来全方位控制松冈的机械手，是非常困难的。在协调多块肌肉的运动时，即使是完成最简单的任务，也需要很复杂的代码。

然而我们的脑就能控制手。可以将脑看做一个性能强大的计算机，如果脑能控制手去系鞋带或是创作艺术，那么计算机程序也一定能。

知道这样的程序存在并不意味着就能找到它们。随着时间的推移，计算机科学家肯定会写出更精深的程序，松冈的机械手将能执行更复杂的活动。这肯定是一个精彩的旅程，但也可能进展缓慢、举步维艰。

一定要这样缓慢吗？想象一下，只要我们简单描述一项任务，马上就会有一个程序提供相应的功能；给计算机输入一段演示人如何打结的电影，然后它立刻就能用机械手重复打结的过程；把莎士比亚全集录入计

算机，然后它就能创作一部新的“莎士比亚”戏剧；只要我们能认出某个东西，就能找到它。这些梦想都能成真——前提是P=NP。

P/NP问题的魅力就在这里。究竟能否让所有的事都变得易如反掌？还是说，有些事情注定就没有简单的解决方法？不能排除这种可能性。无论如何，我们并不指望生活会那么简单。尽管我们并不认为P=NP，但这么美好的世界却让我们忍不住充满憧憬。

1.3 P/NP问题

P/NP问题讨论的是以上所述的所有问题，以及许多与之类似的问题。它们归根到底都是在问：我们搜索大量可能性的速度能有多快？我们找到“金券”（即最佳答案）的过程能变得多容易？

P/NP问题是库尔特·哥德尔在1956年寄给约翰·冯·诺依曼的一封信中首次提出的，哥德尔和冯·诺依曼都是20世纪数学界的泰斗。这封信后来不幸遗失，20世纪80年代又被找到。P/NP问题在学术界的亮相是在20世纪70年代初，由斯蒂芬·库克和列昂尼德·莱文独立提出，当时两位所在的国家正在冷战。之后理查德·卡普列出了这个领域中的21个重要难题，包括前面提到的旅行推销员难题和划分难题。计算机科学家从卡普的工作开始认识到P/NP问题极为重要，由此计算机科学研究的方向发生了戏剧性的转变。如今，P/NP问题的关键性作用已经不仅限于计算机科学领域，还延伸到其他许多领域，如生物学、医学、经济学和物理学。

P/NP问题已成为所有数学领域最难的开放问题之一。1994年安德鲁·怀尔斯证明了费马大定理，受这一消息的鼓舞，克雷数学研究所决定举办竞赛，攻克他们认为最为重要而尚未解决的数学难题。2000年，他们列出了下面这7个千禧年难题，并为每道难题的攻破设立了100万美元的奖金。

1. 贝赫和斯维讷通-戴尔猜想（Birch and Swinnerton-Dyer conjecture）
2. 霍奇猜想（Hodge conjecture）
3. 纳维-斯托克斯方程（Navier-Stokes equations）
4. P/NP问题（P versus NP）
5. 庞加莱猜想（Poincaré conjecture）
6. 黎曼猜想（Riemann hypothesis）
7. 杨-米尔斯理论（Yang-Mills theory）

千禧年难题中的庞加莱猜想已于2003年被格里高利·佩雷尔曼解决，但他拒绝了100万美元的奖金。截至本书写作时其他6个难题都尚未解决。

解决P/NP问题就能拿到100万美元，这可是货真价实的金券啊！

更妙的是，如果你能证明P=NP，那么你也就掌握了找到金券的秘诀，解决其余的千禧年难题将是举手之劳。也就是说，证明了P=NP，你就能解决6道千禧年难题，并得到600万美元。然而证明P=NP或P≠NP可没那么容易。一心想得到600万美元的人最好去玩彩票，那样把握更大一些。

1.4　找到金券

有时候我们能够找到金券。比如我在芝加哥，想开车去纽约，往车载GPS里输入地址，一两分钟之内它就能给出一条从芝加哥到纽约的最快路线，然后我就可以踏上旅程了。几百万字节的内存便可容纳全部的美国街道地图，但地图中可能的路线远远超过几百万。从芝加哥到纽约所有可能的行车路线有多少条？不开错方向的情况下，保守计算可得出的路线数目大到了“不可思议”，即1后边跟63个0。GPS根本没有时间检查所有的可能性，但还是能找到最快的路线。

旅行时间有一个有趣的性质。随便选一个中间地点（比如匹兹堡），从芝加哥经过匹兹堡到纽约的最短路线，一定是芝加哥到匹兹堡的最短路线，再接上匹兹堡到纽约的最短路线。不走匹兹堡可能有更快的路线，但是芝加哥到纽约的最短路线，绝不会比从芝加哥经过匹兹堡到纽约的最短路线还要长。

GPS的计算机程序正是利用了这个性质，快速缩小搜索范围并找到了最佳路线。这可能仍需要检查上亿条路线，但是GPS的计算机处理器完全能胜任，毕竟这个数字比“不可思议”小多了。

找到最短路径并没有体现P/NP问题的全部力量。最短路径问题告诉我们，全部的可能性数量特别大，但这并不总意味着必须遍历所有的可能性才能找到答案。P/NP问题其实是问，是不是对于任意给定的搜索问题，我们都不必遍历所有的可能性就能找到答案？

1.5　漫漫长途

本书讲的是P和NP的故事。什么是P和NP？P/NP描述的是哪类问题？所有搜索问题中最难的问题——“NP完全问题”是怎么回事？这些问题如何影响P/NP问题？

举个简单的例子，Facebook上的好友圈子（即团，clique）中，最大的包含多少人？100人，还是1000人？即使你拥有Facebook的全部数据，这个问题也很难求解。求解较大规模团问题的困难程度，不亚于任何搜索问题。

如果P=NP会怎么样？那么我们将迎来一个美好的世界，计算所有的事物都将易如反掌。我们能快速地了解一切，揭开世界上所有事物的神秘面纱，从治愈绝症到洞悉宇宙的本质。美好的世界也有它灰暗的一面，人们将丧失隐私、丢掉工作，因为没有什么是计算机不能知道或完成的。

这样美好的世界几乎不太可能。困难的搜索问题仍将存在，我们想要甚至需要找到它们的答案。其实我们大可不必放弃。计算机科学家已研发出许多技术，包括很有可能对许多问题奏效的启发式方法，以及能给出接近理想解的近似技术。

P和NP问题是如何产生的？这个故事发生在世界因冷战被割裂的那段日子，其实可以把它分成两个故事来讲。有关高性能计算的思路和问题分别在两个世界独立发展，而这两个世界的研究最终殊途同归，从而产生了P/NP问题。

从哪里着手证明P≠NP？库尔特·哥德尔证明数学不能解决所有的问题。能否用类似的方法，证明存在不能快速解决的搜索问题？为了分析问题的复杂度，我们可以把计算过程分解为最基本的单元。算术几何学（数学的一个抽象分支），为人们能在有朝一日解决这个重要的问题带来了新的希望。但我们距离那一天还很远。

P≠NP会带来什么好处呢？它能帮助我们保守秘密，产生看上去真随机的伪随机数。

未来基于量子力学的计算机能否让P/NP问题变得无足轻重？不太可能，不过量子计算机的建成将解决一部分现在计算机束手无策的问题，比如大数的因数分解。此外，量子力学也会透露P是否等于NP的玄机。

未来将会如何呢？我们仍将面临计算领域的巨大挑战。人们如何与互相协作处理问题的计算机打交道？如何分析每天产生的海量数据？所有事物都能联网，世界将会怎样？要解决这些问题，P/NP问题只会变得更为关键。

1.6 划分难题的解

以下38个数

14 175, 15 055, 16 616, 17 495, 18 072, 19 390, 19 731, 22 161, 23 320, 23 717, 26 343, 28 725, 29 127, 32 257, 40 020, 41 867, 43 155, 46 298, 56 734, 57 176, 58 306, 61 848, 65 825, 66 042, 68 634, 69 189, 72 936, 74 287, 74 537, 81 942, 82 027, 82 623, 82 802, 82 988, 90 467, 97 042, 97 507, 99 564

可分成如下两组：

15 055, 16 616, 19 390, 22 161, 26 343, 40 020, 41 867, 43 155, 46 298,57 176, 58 306, 65 825, 66 042, 69 189, 74 537, 81 942, 82 623, 82 988,90 467

和

14 175, 17 495, 18 072, 19 731, 23 320, 23 717, 28 725, 29 127, 32 257, 56 734, 61 848, 68 634, 72 936, 74 287, 82 027, 82 802, 97 042, 97 507, 99 564

每组中所有数字之和都是1 000 000。

第2章　美妙的世界

假设有人请你写一篇关于过去20年互联网带来的社会变革的论文。你是写装在口袋里的设备让人们能够随时访问所有公开的信息，还是会探讨发生在音乐、电影、出版和新闻产业的巨大变化？无论怎么写，短短一篇论文都很难公允地描述过去20年发生的变化。好了，再想象一下：如果是让你在这些变化尚未发生的20世纪90年代写这篇论文，你又会对未来做出怎样的预测？

如果P=NP被证实，即我们掌握了一种对所有NP问题都有效的快速算法，那么社会将发生更加巨大的变化，互联网看起来只是历史的一个补充说明。不光不可能描述全部的变化，甚至不太可能预测新技术带来的深远影响。

为了能让读者对那个美妙的世界有些许认知，现在让我们来想象一下发现解决NP问题的有效算法几年之后会发生什么。让我们一起跳到2026年，去看看一个P=NP被证实的世界——当然，这个世界是科幻的。首先，咱们来看看这个世界是怎么打造出来的。

2.1 厄巴纳算法

2016年，捷克数学家米莱娜·帕维尔用邮件转发了一个她认为理论上能有效解决NP问题的算法。计算机科学和数学界经过仔细验证，一致认为米莱娜所言不虚，算法确实解决了伟大的P/NP问题。米莱娜将算法发表成论文，文章本身的名字很低调，叫做“论一个库克没有解决的问题”（On an Open Problem of Cook），而《纽约时报》介绍文章的标题更为直白，恰如其分地表明了它的价值：“P=NP”。

2018年，米莱娜成为第一个获得数学领域最高荣誉（菲尔兹奖）的女性。一年后克雷数学研究所奖给米莱娜一张价值100万美元的支票。她成为继格里高利·佩雷尔曼之后第二个成功解决千禧年数学难题的人。不像佩雷尔曼，她高兴地收下了奖金，并将其中一部分（数额未知）捐给了故乡布拉格的大学作为奖学金。

虽然米莱娜的算法在理论上是一个突破，但因为实际执行时间过长而缺乏实用价值。2017年，俄罗斯计算机科学家明斯克·波洛夫采用一种巧妙的方法改良了米莱娜的算法，从而大幅改进了算法的执行效率，然而还是不够实用。

一年后在中国，清华大学的一群本科生仔细优化了明斯克的代码，在当时世界上最快的超级计算机上运行它。他们在几天之内就解决了较大规模的团问题和其他几个常见的NP问题。清华大学开始与波音、戴姆勒-奔驰等工业巨头签约，帮助这些公司解决一些棘手的优化问题。他们帮助波音为新797客机设计出了更好的机翼，使得客机能直接从伦敦飞到悉尼。

史蒂夫·弗兰克是伊利诺伊大学的博士生，此时在清华做一学期的访问学者，他有幸参与了项目的研究工作。回到伊利诺伊州的厄巴纳后，他向导师诉苦，无论他们如何优化，还是要花几天时间才能解决一个中等规模的NP问题。

“如果你遇到灯神，它能满足你一个愿望，你会许什么愿？”导师说。

“不知道。”史蒂夫回答。

“让它给你一个能满足你所有愿望的灯神！”

史蒂夫脑中一亮。他知道一定存在一个能更好解决团问题的算法，而他自己想不出这个算法。但是他遇到了灯神——清华的代码，他可以用很快的速度搜索指数级数量的可能算法。所以他基于清华的例程写了一个搜索更好的NP问题算法的程序。他获准可以使用位于伊利诺伊大学的国家超级计算应用中心（NCSA）的计算资源。经过几周的计算，他的工作初见成效。他找到了一个比清华的算法性能高5%的算法。这一结果发篇论文绰绰有余，但不会产生轰动效应。

他的导师淡淡地说：“用新算法再试一次。”

于是史蒂夫用他的新算法找到了一个更快的解决NP问题的算法。几周后他获得了20%的性能提升。

但他的导师还不满意：“再试一次。”

史蒂夫这样回答：“我为什么不编个程序让计算机自动使用找到的算法来查找更好的算法呢？”

导师的目光变得异样起来，仿佛在说你终于开窍了，又仿佛在说，这么明显的事你怎么现在才想到。

史蒂夫回到办公室，开始写一个算法，让计算机能自动利用搜索到的更快的算法去查找比当前算法更快的算法，这样一直找下去，直到性能无法提升为止。史蒂夫的一个同事问他是否担心天网效应。

“天网是啥？”

“当计算机变得足够聪明，有了自我意识，它就会接管世界，就像《终结者》系列电影里的超级计算机‘天网’一样。”

“不，只是计算机代码而已，不用担心。”

史蒂夫写好了代码，最后一次在NCSA的超级计算机上运行它。随着迭代次数的增加，由计算机自动生成的求解NP问题的算法性能变得越来越好，最后在停止时，生成了一个有4200万行机器码的让人费解的程

序。它求解NP问题的速度，那是相当快。（顺便说一句，计算机依然没有自我意识。）某高校出版社将这个算法自豪地称为“厄巴纳算法”，后来这个名字被保留了下来。

伊利诺伊大学的数学家们开始率先使用厄巴纳算法，用它来证明剩下的千禧年数学难题。证明过程是计算机生成的，复杂到人类几乎无法理解。克雷数学研究所很快发表声明，不再接受基于米莱娜·帕维尔发现的百万美元算法衍生出来的任何算法的数学证明。

许多试图以许可证和买断方式获得厄巴纳算法的公司都陷入了诉讼泥潭，因为与算法有关的各方——资助米莱娜研究的捷克政府、史蒂夫·弗兰克、史蒂夫的导师以及NCSA，都主张自己对厄巴纳算法的所有权。

世界贸易组织在认识到该算法的重大意义后，要求公开厄巴纳算法，供全人类共同享用，相关各方将得到合理的补偿，并成立专门委员会落实。2019年10月23日，厄巴纳算法对所有人公开了。

从此，世界发生了一系列戏剧性的变化……

2.2　计算机1，癌症0

海伦的医生走进检查室，关上身后的门，说：“我有个坏消息，你患有胰腺癌。”

海伦倒抽一口冷气。她才43岁，家里有三个孩子，最大的15岁，最小的才6岁。“你怎么可能知道，我不过验了个血。”

“我们现在只需要得到血液中的标记物和DNA，就可以判断你是否患有癌症，是哪种癌症，以及癌变的程度。如果你坚持，我们可以做活检，但是没必要冒这个险，现有的检查已经足够精确了。”

“我的表姐在8年前被诊断出胰腺癌，那会儿是2018年。当时这病没什么治疗措施，几个月后她就过世了。”

“幸好医学在过去十年里有了很大的进步。我们意识到适合所有人的抗癌方法疗效有限，人们需要针对个体的治疗方案，就是先鉴别出某个人的正常DNA和癌细胞的突变DNA，然后制造出特定的蛋白质，其折叠方式不仅能有效地饿死癌细胞，而且对正常细胞没有任何影响。死亡的癌细胞会被排出体外。”

“听起来会很贵。”海伦评论道。

“化疗才贵呢，这种疗法的成本只有几千美元，将来还会更便宜。大部分费用都可以由你的医疗保险支付。”

“太好了！过去十年究竟发生了什么，让检验和治疗变得如此简便？”

“这些想法很早就有了，但在过去，即使是世界上最快的计算机在解决DNA密码方面也显得太慢，所以一直没有什么突破。厄巴纳算法的出现改变了这种状况，过去几年里我们取得了难以置信的进步。通常一项新技术出来后，我们会做很多年实验才会投入临床使用，但是这种疗法的初期测试太成功了，以至于FDA觉得不立即批准投入使用都不道德，毕竟有那么多的癌症患者，他们等不起。”

“什么时候开始治疗？”海伦问。

“开始？我们已经用抽的血做完了分析，这是你的处方。”

医生递给海伦的不是一张纸，而是一个U盘。“这里面有治疗你的癌症的蛋白质的编码信息。把它带到药房，让他们为你制作药片。一天一片，连吃两周，你体内的癌细胞就清扫干净了，没有任何副作用。但是要注意，这个处方只对你起作用，其他人的DNA不同，如果吃了你的药片，可能会产生严重的后果，甚至是致命的。”

“没有副作用？头发不会掉光？没有恶心想吐的感觉？癌症及早发现的话只要服几片药就好了，跟治感冒一样？所有这一切都得益于一个算法？”

“不完全准确，”医生说，“没错，厄巴纳算法帮人们战胜了癌魔，治愈了艾滋病和糖尿病。可是，我们还不知道如何应对普通感冒。”

2.3 棒球比赛

“真是打棒球的好天气！”兰迪对他12岁的女儿凯特说，这是他第一次带女儿到圣路易斯看棒球比赛。圣路易斯红雀队主场对战密尔沃基酿酒人，这场2026赛季末的夺冠赛扣人心弦。兰迪感叹棒球运动和他小时候相比变化不少。这项运动在科技的使用上曾经是那么地原始，甚至连计时装置都不用，而如今它同样受到了厄巴纳算法的巨大影响。

首先当然是棒球赛程安排上的变化。直到2004年，棒球大联盟（Major League Baseball）的赛程都是由亨利和霍莉·斯蒂芬森夫妇来安排的，这对夫妻档安排赛程的依据是几条简单的规则，例如每个队主场和客场的比赛数最好大致相等，还会照顾某些地方的偏好，如波士顿的球迷希望把波士顿红袜队的一次主场比赛安排在四月中旬爱国者日的白天，因为波士顿马拉松赛的参赛人员会在那天从球场旁边经过。2005年棒球大联盟与匹兹堡的赛事筹划集团（Sports Scheduling Group）签订了合作协议，因为这家公司知道如何避免两支队伍在连续的几周内多次交锋。众所周知，暴雨天气无法进行室外棒球赛，但赛事筹划集团在安排赛事时从不考虑下雨的因素，原因很简单，没办法在开赛前的12月份预测整个赛季的天气状况。当然，那是厄巴纳算法出现之前的事情了。

在算法的帮助下，人类的天气预测技术取得了不可思议的进步，可以提前一年准确预测温度、风力、云量和降水。类似的算法可以做到准确预测风暴、龙卷风和飓风，让人们有充足的准备和撤离时间。新的天气预测技术同样改变了人们的日常生活。学校会对未来下雪的日子提前做准备。承办户外婚礼的教堂会根据未来婚期的天气状况向新人收取不同的费用，想在好天气结婚的要多交钱，愿意在炎热、潮湿或下雨天举行露天婚礼的则可以享受折扣。大多数人都想在好天气看棒球赛，所以如果底特律会下雨或者阴天的话，完全可以把比赛安排到明尼阿波利斯去，那里将艳阳高照。

赛场和电视机前的观众都想看到高质量的比赛，尤其是临近赛季末尾时优秀球队之间的激烈比拼。兰迪小时候，判断哪支球队会赢没有科学的方法，基本靠猜。而新的预测工具料事如神，能提前告诉人们哪些球队将进入9月份的冠军争夺赛。计算机预测胜场也并非完全准确。棒球运动具有随机性，每个赛季的实际战局都会与预测有那么几场的出入。然

而算法的模型仍能以极高的准确率预测哪些球队会在赛季末尾名列前茅。那些没被预测为夺冠热门的球队的球迷们每年都会抱怨计算机算错了，然而计算机通常是对的，唯一的一次预测失败，被人们津津乐道。

棒球产业的大亨们想用更好的方式安排比赛，目标之一是让每个人都能在最好的天气看到强队之间的火拼，当然还有更实际的成本控制方面的诉求，比如尽可能缩短每个团队花在路上的总时间。15年前，根本不可能将很多方面纳入考虑，那会使需要检验的计划规模庞大得难以处理。但现在，无论要求多么苛刻，厄巴纳算法都能在几分钟内给出一个最棒的赛程表。

所以兰迪才能在这样完美的天气带凯特去看这样一场关键的棒球赛。虽然这天红雀队的观赛费用比平时高，但兰迪很乐意为此买单。

兰迪和凯特进入看台的时候没有检票员和检票闸机，他们就直接走进了体育场，监控摄像头通过人脸识别软件认出了他们，通过数据匹配证实他们已经买过票了。如果有人企图不买票就进场，保安肯定会找出他，人们很早就知道了这一事实，不敢冒险去挑战识别软件的威力了。

兰迪和女儿就座后，兰迪四处打量，发现偌大的球场空空荡荡，看不到几年前巨大的电视摄像机和操作员了。取代这些的是围绕在球场周围的20台小得几乎看不见的摄影机。20台摄像机的画面输入计算机后，软件利用厄巴纳算法构建出比赛的实时三维渲染数据。从这些渲染数据中，计算机可以合成从任意角度拍摄球场任意地点的视频短片。观众能从捕手的视角观看比赛，也能从跑垒员的视角观看他滑向三垒的英姿。虽然这些画面其实是计算机合成的，但看起来非常逼真。在一次有名的实验中，请100个观众观看并排放在一起的两个短片，其中一个短片是实际拍摄的，另一个是计算机合成的。结果89人都误以为计算机合成的短片是实际拍摄的。

新型电视允许人们直接下载三维渲染数据，然后控制虚拟的摄像机飞到赛场上的任意位置，随心所欲地享受比赛过程。

计算机在球棒击球的那一瞬间，准确预测出球的飞行轨迹，然后立即选一个最佳的观看角度，把比赛实时呈现出来。运营电视网络转播的只有四个人：两个评论员，一个制片人，还有一个技术员（处理偶发的故障）。评论员说的是英语，而由于圣路易斯和密尔沃基两队分别有一个

知名的日本球员，这场比赛在东京也很受关注，那里的观众听到的将是日语的评论。实际上日本观众听到的是美国的评论员在说话，但是计算机立即用语音识别、翻译和语音合成工具模拟出地道的日语。这些工具自然也使用了厄巴纳算法。这场比赛在全世界用876种语言和方言播出。

转播一场比赛用四个人似乎已经很少了，但在电视网络的副总裁看来，最好一个人都不用。几年以来，大学校园都通过布置在球场周围的摄像机记录练习赛，供教练们进行战术分析。最近，一个具有创业精神的学生成功地将厄巴纳算法应用到这些录像数据上，制作出全套的转播视频，其中评论、特写视角、数据统计样样俱全，一切完全由计算机自动生成。很快所有大学（以及许多高中比赛和小型联赛）的每种运动赛事都可以在互联网上实时观看，还拥有各种语言的版本。这些转播的质量也许比不上真人做评论的那些节目，但也差不到哪里去。

幻想比赛达到了前所未有的高度。计算机可以生成体育迷们梦想中看到的比赛，不仅可以让不在一支球队的球星们并肩作战，甚至还可以关公战秦琼，让活跃在不同时代的球星同场竞技：让全盛时期的传奇投手赛·杨对战如日中天的乔·狄马乔，或是1927年的洋基队对战1998年的洋基队。幻想与现实变得难以区分。某个愚人节，ESPN电视台的一个计算机程序员改写了部分代码，让电视机前的观众看到他所喜爱的波士顿凯尔特人队打败了纽约尼克斯队，而现实比赛中，尼克斯才是真正的胜者。这次转播事故让许多人丢了饭碗。

计算机充当裁判也表现完美，它总能精准无误地判别坏球、好球、出局和全垒打。为了削减成本，小型联赛和大学里的所有比赛都用计算机来做裁判。而大型联赛坚持使用真人裁判，每当错误的判决改变比赛的局势都会遭到一片质疑之声。

由于担心计算机做管理工作也比人类经理更为出色（这种担心是正确的），棒球大联盟在经历过2022年世界巡回赛的失败之后，宣布禁止球队使用计算机设备。负责人称：“这是为了棒球运动好。”

兰迪一边喝着啤酒一边啃着汉堡（饮料和食物变得更好吃了，因为厄巴纳算法改进了配方），一边还欣赏着比赛。比赛进行的方式和他小的时候相比没有什么变化。他其实完全可以省下大笔钱，在家里欣赏从完美

视角呈现的比赛三维实况转播。尽管科技让在家看球的体验远比在体育场看球还要好，兰迪还是愿意享受一种参与感，厄巴纳算法再厉害也无法再现这种感情上的依恋。

计算机可以自动生成计分表，但兰迪还是教女儿凯特怎样在纸上计分，就像他父亲教他的那样。纵使以厄巴纳算法为代表的科技改变了几乎所有的事物，但比赛还是那个比赛。三振出局的老规矩，永远都不会改变。

2.4 奥卡姆剃刀

为何形式简单的计算机代码，只要强大到厄巴纳算法那种级别，就能带给我们一个理想的世界，让人们可以治愈疾病、预测天气，以及创造几乎能够以假乱真的虚拟环境？为了回答这个问题，我们需要回到14世纪的早期。

奥卡姆的威廉还是一名青年时就加入了英国的方济各会。和许多14世纪的宗教团体一样，方济各会的总部设在牛津，它为牛津大学的在校生提供了住所。威廉是一名主修神学的牛津学生，尽管没有毕业，他后来还是成了中世纪最主要的思想家之一，对物理学、神学、逻辑学和哲学都作出了巨大贡献。他最著名的思想是奥卡姆剃刀法则，即主张最简单的解释通常是最好的解释。“剃刀”一词比喻我们可以把一个理论中纠缠复杂的部分像胡子一样“剃除”，只留下最简单的解释。这一思想贯穿文艺复兴时代一直延续到今天，引领着一代又一代人的科学和哲学思潮。

17世纪法国哲学家勒内·笛卡儿通晓奥卡姆剃刀法则。他怀疑一切，甚至对周围世界的存在性提出质疑。笛卡儿最为著名的哲学命题是Cogito ergo sum，即“我思故我在”。在他的著作《方法论》中，笛卡尔摒弃了一切前提假设，仅从他能通过思考意识到自己这一点，得出了“笛卡尔是存在的”这一结论。那么笛卡尔如何解释他感受到的复杂世界呢？万物是否有可能只存在于笛卡尔的意识中？笛卡儿不赞同这种想法，因为有另一种更简单和更合理的解释，那就是其他人和他一样，大家共同生活在一个可以认知和理解的物质世界之中。

> 我被“我思故我在”这几个字迷住了，我确信它是真理，证据只有一条，那就是我清楚地看到，存在是一个事物能够思辨的必要前提。我甚至可以论断，一般地讲，一切我们能清晰明确地在脑中显现的事物都是真实的。然而值得注意的问题是，很难恰当地分辨哪些物体是我们能够明确地想象的。
>
> 笛卡儿，《方法论》，第四部分

和笛卡儿同时代的约翰内斯·开普勒通过观测行星运动，归纳出了一系列描述它们运动路线和速度的定律。至于为什么行星运动符合这些定

律，开普勒无法给出一个简单的解释。

艾萨克·牛顿将奥卡姆剃刀法则运用到物理世界。他著名的运动定律以非常简单的方式，表述了物体如何相互作用：

1. 如果不给物体施加任何力，静止的物体会保持静止，运动的物体速度保持不变；
2. 施加于物体上的力将产生一个与其大小成正比的固定加速度；
3. 一个物体给另一个物体施加力，则施力物体也会受到来自受力物体的大小相等、方向相反的力。

通过把运动定律和他对引力的简单描述相结合，牛顿展示了如何从天体的运动推导出开普勒定律。简单的表述可以具有强大的解释能力。

几个世纪后，阿尔伯特·爱因斯坦和其他科学家从理论上提出：对于运动速度接近光速的物体，牛顿的简单运动定律是不成立的。后来的一些实验基本证明爱因斯坦是对的。他的名言是："把所有事都变得尽可能简单，而不可过于简单。"这并不意味着牛顿错了，恰恰相反，他的模型是对我们所认知的世界的一个很好的近似。甚至在今天，对于包括驾驶汽车，以及高中大部分科学实验在内的一些简单场合，牛顿定律仍然非常适用。

对于非常小的粒子，甚至爱因斯坦的理论也不再成立，它们所服从的法则被称为量子力学。当代物理学家试图找到一种能融合爱因斯坦的广义相对论和量子力学的理论，即所谓的"大统一理论"。

简单的模型永远无法反映世界全部的复杂性，但它们通常可以很接近这个目标。为某些情形找到一种简单的解释，就能对未来发生的类似情形做出很好的预测。近年来我们看到这个原理在计算机科学世界得到了充分的体现。

今天我可以手写一张支票，用手机拍张支票的照片，然后通过互联网存入支票。银行的软件通过照片就能识别交易金额和账号，即使支票是手写的也没关系。如果没有疑议，银行职员甚至都不用亲自去看这张支票。

识别位于支票底部的账号比较容易，组成账号的数字符合固定的格式，

这是特意为方便计算机识别而设计的。

但是支票上的金额是手写的“30美元”。每个人写字的方式都不一样，计算机如何能判别这张支票的金额呢？

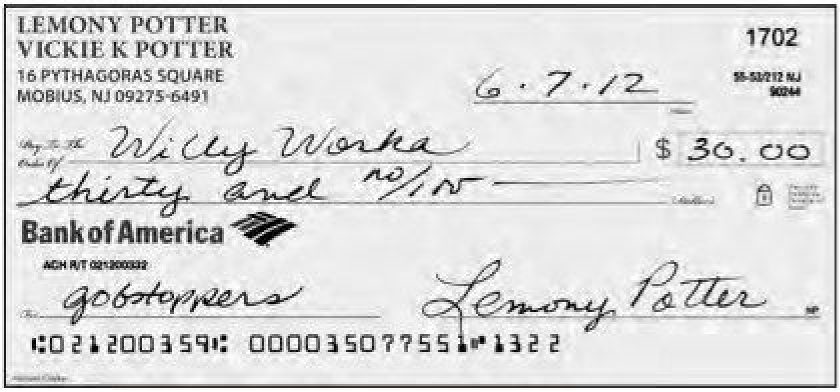


图**2-1**　支票

这个问题不简单。不信你看数字2有多少种写法。

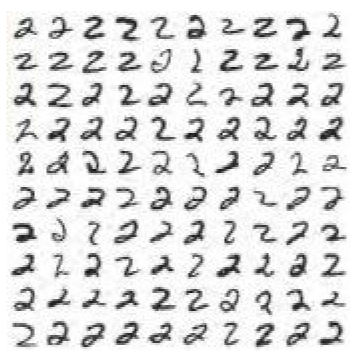


图**2-2**　各种手写的数字**2**

计算机科学的一个分支——机器学习就是处理这类问题的。首先用大量的数据训练一个算法，在这个例子里，数据就是人们写数字2和其他数字的样例。算法会建立一个相对简单的模型，然后尝试用这个模型正确区分各种手写的数字。训练得好的算法能对新的数字进行正确的分类，即使是对训练期之后写出的数字也不例外。

过去20年，计算机科学家在机器学习领域取得了长足的进步，研究出了通过检查几千到几百万样例，计算出将数据分类的正确方式。除了识别支票之外，许多照片处理程序还可以根据人脸来对照片排序，效果还过

得去。Amazon、Netflix和Pandora等公司能为顾客推荐图书、电影和音乐，根据的是顾客之前购买过的商品，以及观影和听歌的模式。语音识别和语言翻译虽然不能做到和真人一样，但会让你大致了解写的或说的是什么。垃圾邮件识别工具屏蔽了人们不想要的几乎所有信息。到2020年，所有的汽车有望能够自动驾驶。

这类技术所能做到的差不多就是这些了，未来取得的进步最终将越变越小。这是否意味着奥卡姆剃刀不再锋利，宝刀已老？

倒也未必。奥卡姆剃刀法则说的是最简单的解释问题的方法就是最好的方法，但它并没有告诉你如何找到那个最简表述。如今的机器学习技术只能使用简单的数据表述方式，通常会把数据的多种特征用相对直接的方式加以合并。如何找到最简表述，即找到任意形式的程序中最短小而有效的数据分类程序，是一个困难的问题，一个NP问题。

如果能使用快速解决所有NP问题的厄巴纳算法，找到最简单的数据分类程序就变成了一道简单的编程练习。我们需要做的只是把海量数据交给算法，然后看着它完成剩下的工作。这样一来，我们几乎能了解所有的事物。

我们已经看到了厄巴纳算法帮助人类治愈了很多疾病，并且改变了美国的棒球运动。接下来让我们回到未来，看看它是如何改变艺术的。

2.5 创造力的自动化

借助奥卡姆剃刀，厄巴纳算法几乎能让我们了解一切，包括艺术作品为何优秀，音乐为何流行，言语为何动人心魄。要知道P=NP意味着只要能评测一个事物，我们就能找到它。所以只要有一个慧眼识珠的算法，再用厄巴纳算法就能很快找到这颗“明珠”。

2022年，美国民主党总统候选人初选即将在科罗拉多州开始。之前的两周，皮特·约翰逊得票第三，远远落后于前两名。情况在“那次演讲”后改变了。在韦尔市的一个小剧场里，约翰逊进行了一次短短10分钟的演讲，甚至没有媒体列席。然而这次有关科罗拉多和美国的演讲格外动人心魄，在场的32个人起立报以热烈的掌声。

一个与会者用手机录下了演讲，发到了网上。几百万人在线观看了演讲，后来甚至有几千人言之凿凿，说自己亲临了那次演讲。约翰逊赢得了初选，成为2024年总统竞选的热门候选人。为进一步提高自己的曝光率，皮特·约翰逊炒了自己的竞选经理，然后换了一个享誉全国的专家。

被炒的前竞选经理忿忿不平，召开记者会爆料了那次演讲背后的真相。当时情况不妙，如果不马上使出一些厉害手段，皮特·约翰逊的竞选之旅眼看就要到头了。于是，竞选经理雇了一个计算机程序员，程序员先下载了数以万计的近几十年来广受欢迎的演讲，然后用厄巴纳算法生成了一个关于科罗拉多及国内外局势的优秀演讲稿。皮特·约翰逊经过多次排练，把计算机生成的稿子背得滚瓜烂熟，然后才有了韦尔市的演讲。竞选经理请观众把此次记者会的录像也传到网上。

舆论一片哗然。有些人表示愤慨，另一些则认为政客们的演讲稿是由职业撰稿人捉刀还是由计算机生成没多大差别。皮特·约翰逊失掉了民心，也失掉了大选。到2026年的时候，几乎所有候选人撰写演讲稿时都会让计算机帮忙甚至代劳，大家习以为常，见怪不怪。当然，人人演讲出色，等于没人演讲出色。

古典音乐家用厄巴纳算法完成大师们未完成的名作，如普契尼的歌剧《图兰朵》、马勒的《第十交响曲》、舒伯特的《第八交响曲》（又名

《未完成交响曲》）；而且还能再续经典，如创作出贝多芬的第十部交响曲。由算法生成的最著名的交响曲叫做《厄巴纳第一交响曲》，古典乐迷非常喜爱它。有人甚至用厄巴纳算法制作出甲壳虫和猫王的新专辑，将已故歌星们的嗓音模仿得惟妙惟肖。音乐评论界的专家对这些“作品”嗤之以鼻，认为它们只是照猫画虎，缺乏真正的创意，但还是有很多人下载。

还有人用计算机生成新的画作、小说、戏剧和诗歌。一个电影爱好者最近制作了一部由盛年的亨弗莱·鲍嘉和茱莉娅·罗伯茨主演的爱情喜剧片，看起来就好像两人真的穿越了时光，在片场大飙感情戏一样。想看由导演蒂姆·伯顿执镜的《绿野仙踪》？没问题！

Amazon能做的超越了原来的图书推荐。你只要花一张电影票的钱，就可以让Amazon写一部完全符合你口味的小说。NBC电视台目前推出了一档“现场直播”的动作冒险剧集，完全由计算机创作，不需要任何编剧和演员。最新的Xbox上的电子游戏带给你沉浸式的叙事和世界体验。故事的发展不再是线性的，而是有着无限的可能性，玩家完全按自由意志行动，并为自己的行为承担相应的后果。不少人觉得难以区分游戏和现实生活，比较明显的区别是现实生活可能没有游戏那么刺激。

这些新的娱乐选择让世人目眩神迷，为之痴狂。许多文章都在批判真实表达的缺失。究竟厄巴纳算法标志着一个艺术新纪元的诞生，还是人类创造力的死亡，这在未来的几个世纪仍是一个争议不断的话题。

2.6 终极侦探

司法机关觉得厄巴纳算法在侦破犯罪案件方面堪称鬼才，它在追捕嫌犯时总是能完成不可能完成的任务。然而也有一些争议。

后厄巴纳算法时代的亚特兰大出现了一起多人死亡的凶杀案，警方从犯罪现场的快餐包装纸上收集到了一份DNA，但在美国DNA联合索引系统（CODIS）中找不到与之匹配的数据。佐治亚理工学院的一个计算机科学家用厄巴纳算法和CODIS创建了一个新的程序，该程序不仅可以利用这些DNA数据预测嫌犯的身高、眼睛和皮肤颜色等外貌特征，还能预测嫌犯的个性特点以及可能从事的行业。然后这个计算机科学家带领学生，把CODIS中的DNA数据和这些人的面部照片匹配，从而研发出一种技术，根据某人的DNA生成相貌的大致素描，前提是知道目标的年龄和面部毛发类型。

于是警方想用这个算法找到和犯罪现场DNA匹配的罪犯头像照片，但没有成功。他们又把算法运行在驾驶执照的头像数据库里，这回找到了一个嫌疑人——乔治·布朗，几小时后把他抓了进来。乔治·布朗没有不在场的证明，而且是唯一符合由DNA生成的相貌素描的人，但除此之外，没有发现他与此次犯罪有明显关联。他拒绝提供DNA样本，警方得到了法院的搜查令，然后比对了他和犯罪现场留下的DNA，果然相符。乔治·布朗被认定有罪，判处死刑。

布朗提出上诉，他的律师说根据犯罪现场留下的DNA画出素描的行为侵犯了乔治·布朗的隐私权。美国最高法院裁定，使用DNA判定一个人的身份是合法的。警方只是将算法运行在合法取得的证据上。

然而，此案引发了公众对丧失隐私权的大规模抗议。38个州决定DNA只能用来匹配合法收录到CODIS中的嫌犯，除此之外不能使用。最后，佐治亚州州长将乔治·布朗的死刑改判为终身监禁。

2.7 美妙世界的阴暗面

以前人们依靠公钥加密技术，可以不经过初始化设置就安全地通过网络将信用卡号传给交易公司。而厄巴纳算法能在瞬间破解公钥加密。起初，电子商务的发展经历了一些挫折，但互联网巨头们很快就联合起来，将业务迁移到使用私钥加密的系统。很快一个私钥注册处被建立起来，人们从街边的药店就能买到一个密钥U盘，里面装着几十亿个一次性的私钥。虽然比原来稍显不便，但是电子商务活动又兴旺起来。

更多人感到个人隐私被剥夺，并对此感到恐慌。原来看起来人畜无害的视频摄像头突然能识别和追踪每个人了。更恐怖的是，计算机算法能准确预测你走哪条路，听什么歌，看什么电影，买什么产品，简直比你自己还了解你。定向投放的广告很容易就能诱导你改变消费习惯。被认为可能是窃贼或恐怖分子的人会立刻被系统认出来，受到严密的监视。

对厄巴纳算法最大的顾虑是就业问题。从秘书到中层管理人员，几乎所有的白领职位都受到算法的威胁，它可以理解各种信息（甚至包括不太正式的电子邮件、音频和视频），生成信件和报表，还能根据分析得出结论和制定决策。那些原来需要外包的工作都交给计算机算法做就行了，这被称为"厄巴纳包"。工作人员如果不能让雇主相信自己提供的附加价值，就会失业。大公司继续理直气壮地削减薪资。不少公司撤销了设在海外的人工咨询台服务，代之以计算机化的支持多种语言和方言的应答系统。许多消费者在民意测验中表示计算机提供的服务比被取代的人工服务更加有用。

有些国家开始立法限制厄巴纳算法的应用以保护就业岗位，但在来自其他国家的商业竞争压力面前，这些法律很快就被废止了。情况逐渐有了一些转好的迹象。厄巴纳算法刺激了疲软的经济，催生了新兴产业的雏形。大学里开设了新的课程，如"厄巴纳优化"，讲授拿到一个问题后如何以最快捷简便的方式运用厄巴纳算法解决它。虽然长远来看，算法创造的就业机将会远远多于失去的就业机会，但还是有许多人对算法耿耿于怀，替那些无法适应新经济体制的失业者鸣不平。

政府继续推行法令，保护民众不受厄巴纳算法的冲击，但科技的影响覆水难收。人类总是能很快适应新形势，渐渐很少有人会在民意测验中表

示愿意时光倒流，回到古老的2012年，逃离这个算法带来的美妙世界。

2.8 回到现实

觉得一个算法能改变一切实在令人难以置信？只要能辨识，就能找到？我们能轻松了解所有事物？确实不太可能，因而大部分计算机科学家都认为P≠NP，即永远不会发现厄巴纳算法或是类似的算法。

无论P/NP问题的命运如何，本章出现的故事中有一部分还是可能会变成现实的。但那将是一场持久战，我们必须脚踏实地，逐个攻破技术难关，而不是只靠神奇的计算机代码去解决所有的问题。人类的创造力是很强大的，只要有梦想在前方召唤，我们最终一定能设法到达。

第3章　P和NP

3.1 敌友国

为了更好地了解P和NP，让我们访问一个假想中的国度——敌友国。在那里，任意两个人之间的关系不是朋友，便是敌人。

敌友国住着2万居民。单看每个人都很正常，但是把两个人放在一起就会发生奇怪的事情。他们见到对方的第一眼，要么立刻变成最好的朋友，要么马上变成最坏的敌人。虽然叫敌友国，但从没看到过两个人的关系会有中间立场，而是非敌即友。

看上去敌友关系几乎是随机建立的。朋友的朋友不一定是朋友，也可能是敌人，敌人的敌人亦然。它和性别、种族、信仰和社会地位都没什么关系，不过人们更倾向于树敌而不是交友，大部分人的朋友数量都要比敌人少很多。

互联网提供了大量敌友国中朋友关系的数据。通过检查Facebook和Twitter，敌友国理工学院的计算机科学家们获得了几乎完整的数据库，里面记录了哪两个人是朋友，哪两个是敌人。在本章，我们将看到研究者能利用这些数据做什么，不能做什么。

3.2　六度理论

随便从敌友国挑两个人出来，不妨叫他俩Alice和George，两个人互为朋友的可能性很低。但他们可能有一个共同的朋友Bob，也可能没有这么一个人。敌友国理工学院的计算机科学家把全国的人标记在一张散点图上（每个点代表一个人），并在代表朋友关系的两点之间连一条线。这张图的一部分像下面这样。

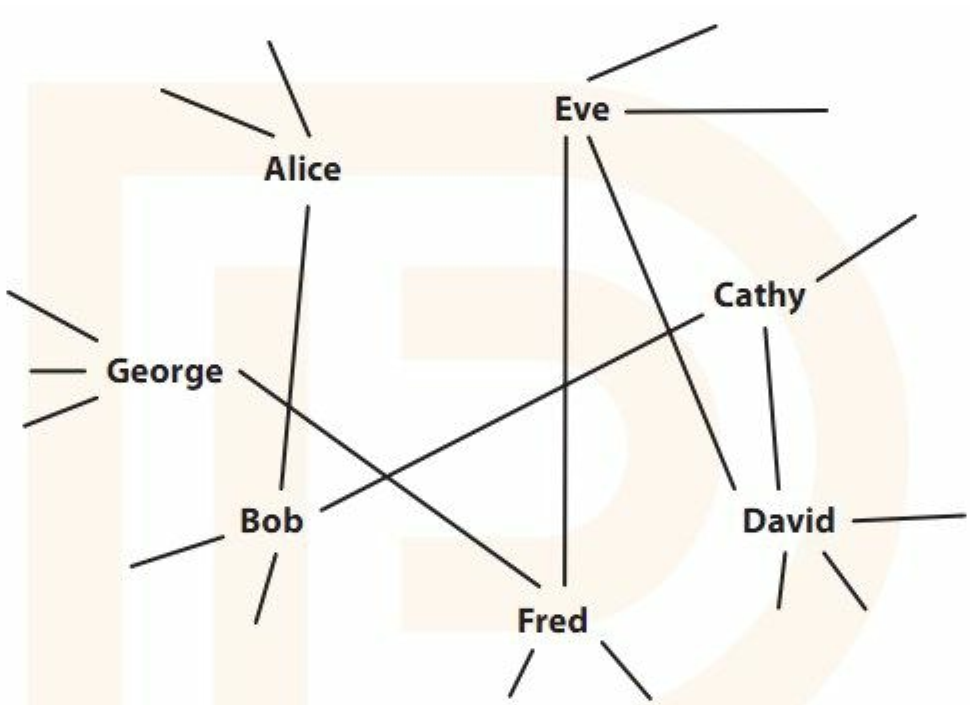


图**3-1**　敌友国的好友关系图

研究者发现要连接Alice和George，中间要经过6个人：Alice和Bob是朋友，Bob和Cathy是朋友，Cathy和David是朋友，David和Eve是朋友，Eve和Fred是朋友，Fred和George是朋友。敌友国的科学家想，是不是任意两个人都能通过一条足够短的好友链联系起来。这个现象称为“小世界”，不是得名于迪士尼乐园里那个游乐项目，而是因为两个陌生人见面，通过攀谈发现彼此竟然七拐八弯地有关联时，通常会说：“世界真小啊！”

1967年心理学家斯坦利·米尔格拉姆做了一个著名的实验来检验小世界

理论。他首先挑选了一个波士顿地区的股票经纪人，真名不便透露，我们不妨叫他汤姆·琼斯。米尔格拉姆随机选择了100个来自内布拉斯加州炒股的人作为第一组，100个不炒股的人作为第二组，另外还有从波士顿登报招募的100人作为第三组。来自内布拉斯加州的第二组人和波士顿第三组的人跟投资行当没什么特殊的联系。米尔格拉姆给每个人一个装有实验指令的档案夹、一本花名册，以及15张贴了邮票能寄给哈佛大学米尔格拉姆的明信片。实验指令是这样写的。

1. 把你的名字添到花名册上。
2. 拿一张明信片，把它塞到邮筒里寄出去。
3. 如果你和波士顿的股票经纪人汤姆·琼斯有私交，请把这个档案夹寄给他。
4. 如果你不认识汤姆·琼斯，把档案夹寄给一个你认识的人，这个人的名字没有出现在花名册上，而且你觉得他更有可能认识汤姆·琼斯。

300个人中总共有217个把档案寄给了自己的朋友，其中有64份档案最终到了股票经纪人那里。好友链的平均长度是5.2个人，从而产生了所谓六度分割的概念，就是说任何人最多经过6个人就能认识所有人。尽管许多人从多方面指出了米尔格拉姆研究的不足，并且他本人从没有声称自己发现了六度空间法则，但是很明显，我们所有人之间的联系比预想中更为紧密。

如果要用某种比仅仅听说过某人更为明确的方式来定义两人之间有关联，可以分析一下现实中的人际网络。这启发了一些休闲时玩的小游戏，基本玩法是看看某人与一个特定的交际很广的人的距离有多远。1994年，演员凯文·贝肯在宣传他出演的电影《狂野之河》（*The River Wild*）时开了一个玩笑，说他和好莱坞的每个人都共事过，或者说，他和某个跟每个人都共事过的人一起共事过。于是大家开始玩“凯文·贝肯的六度空间”这个小游戏，试着找出任意一个演员和凯文·贝肯之间联系的途径，而对两人之间有关联的定义是出演过同一部片子。这样一来，大部分演员和凯文·贝肯的关系链都很短，所以其实他们彼此之间也很近。例如，从查理·卓别林到凯文·贝肯的关系链长度为3：卓别林执导了1967年的电影《香港女伯爵》（*A Countess from Hong Kong*）并在其中扮演了一个小角色，女伯爵由索菲亚·罗兰扮演；索菲亚·罗兰主演了1979年名不见经传的电影《火力》（*Firepower*）；伊莱·瓦拉赫是《火

力》的主要演员之一，在《神秘河》（*Mystic River*）中他扮演了一个跑龙套的小角色，而凯文·贝肯也参演了该片。

数学家们也有一个类似的游戏，通过“曾经共同撰写一篇论文”这种关系，看看某个数学家离发表著作很多的组合数学家保罗·爱多士有多远。[1]

1 我和三个与保罗·爱多士合著过论文的人一起写过论文，所以我的“爱多士分数”是2。爱多士于1996年辞世，所以我不太可能刷新这个分数了。我没有演戏的经历（或天赋），所以没有“贝肯分数”。

敌友国的研究者决定首先检验六度空间法则在他们的国家是否成立。要检验Alice和George之间是否有长度为6的关系链，一个简单的方法是检查所有长度为6的关系链，看看其中是否有一条以Alice和George为端点。但是对于敌友国的2万居民，所有长度为6的关系链有3 198 400 279 980 000 480 000种可能，即使计算机每秒能检查1万亿条关系链，也需要100多年才能检查完。有没有更好的方法来检查Alice和George之间的距离呢？

有。可以用几个简单的步骤来快速找出Alice和George之间的距离有多远。

- 把点Alice标为0。
- 把所有Alice的朋友的点标为1。
- 遍历所有标记为1的点，把每个点的朋友标为2（除去那些已经标记了点的人）。
- 遍历所有标记为2的点，把每个点的朋友标为3（除去那些已经标记了点的人）。
- 如此进行，直到点George被一个数字标记。
- 点George上标的数字就是Alice和George之间的距离。

像这样对计算步骤做出的非正式描述就叫算法（algorithm），得名于19世纪波斯数学家穆罕默德·伊本·穆萨·花剌子密。公元825年，花剌子密发表了《论印度数字的计算》（*On the Calculation with Hindu Numerals*），这本著作的意义是将印度的计数系统推广到中东和欧洲。书名的拉丁文译法是*Algoritmi de numero Indorum*，Al-Khwārizmī（花剌子密）在拉丁语里变成了Algoritmi，然后变成了今天的术语algorithm。

上述算法能用差不多50万步计算出Alice和George的距离。要找到所有敌友国居民中任意一对之间隔了多远，我们需要更高明的算法。Floyd-Warshall算法能用大概8万亿步就计算出所有这些距离。可能你觉得1万亿这个数听起来挺大，但如今的个人计算机每秒都能执行几十亿个操作。敌友国理工学院的机器在几分钟内就计算出了全部国民之间的分隔距离。科学家们发现本国的平均分隔距离比6要稍大一点，尽管有些人组成的朋友圈非常孤立，里面的人和圈外的人都不是朋友。

我们可不能低估刚才发生的事情的意义。当然研究者可以编程遍历所有路径，找出其中连接Alice和George且最短的一条，但这样做可能要检查的路径太多了，导致程序没法在合理时限内跑完。然而采用某种更简单的算法，他们能在一秒内算出Alice和George之间的距离，并且在几分钟内算出敌友国居民中任意一对之间的距离。

3.3　牵线搭桥

在敌友国，一段美满的姻缘几乎完全取决于双方是否是很好的朋友。然而敌友关系的分布完全没有规律，有些幸运儿可能碰巧找到了合适的人选，剩下的人则很难找到一个好伴侣。

敌友国的研究者意识到可以用他们的数据来为社会服务，让成功婚姻的数量最大化。于是他们在网站上招募了200个志愿者，其中一半是男性，一半是女性，所有人都是异性恋。他们想尽可能多地为合适的男女牵线搭桥。

一共需要搜索多少种可能性呢？和第一个男性配对的女性人选有100个可能。做出选择后剩下的99个女性可能和第二个男性配对，然后是98个女性和第三个男性配对，如此等等。这样所有的可能数是100乘以99乘以98乘以……乘以2乘以1，这个值被称为“100的阶乘”，写作100!，这个数有158位，比googol还大。googol这个名字是数学家爱德华·卡斯纳九岁的侄子起的，当时卡斯纳想给1后边跟100个0的数起名。

互联网公司Google得名于对googol的错误拼写，它是对该公司搜索引擎所处理的海量数据的一个形象但非常不准确的表示。互联网这种庞然大物（实际上无法测量它有多大）所包含的信息量无论分得多细，都还远远达不到googol这个数量级。所以，即使动用全部的计算机算到地老天荒，也不可能搜索googol数量级的信息，更别提100!了。

然而敌友国研究者还是能找到成功配对数的最大可能值，只需换一个高明的算法即可。下边这张图显示了哪些男女志愿者是朋友关系。

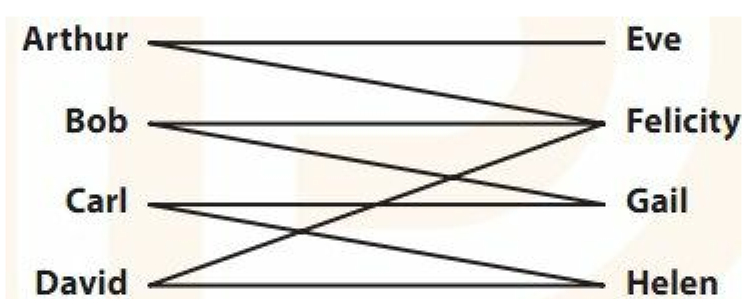


图**3-2**　敌友国的男女们

让我们看看谁跟谁可能是天作之合。首先我们把Arthur和Eve凑成一对。Bob和Felicity这对朋友还没有在一起，让我们促成他俩，同样祝福Carl和Gail。现在我们有了一张新图，虚线表示我们做出的配对选择。

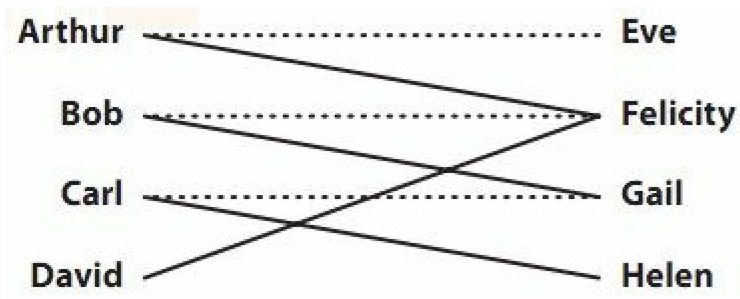


图**3-3**　部分配对的男女们

这下不存在能够配对的朋友了，这是可能的配对结果中最好的吗？不一定。

David没有伴侣，但他和Felicity是朋友，而Felicity和Bob在一起。Bob和Gail是朋友（但我们没有撮合他俩）。Gail和Carl是一对。Carl和没有伴儿的Helen是朋友。如果我们拆散Bob和Feliciy，再坏了Carl和Gail的好事，然后重新配对，那么这下所有人都成功配对了。

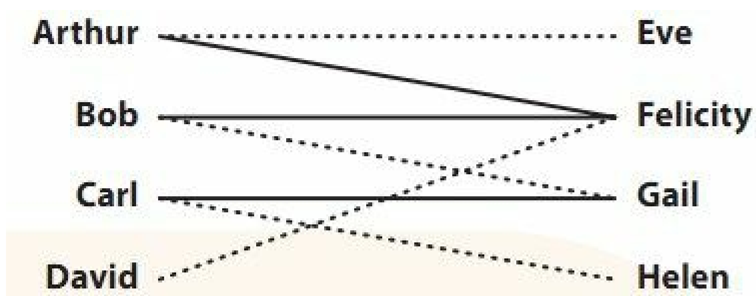


图**3-4**　全部配对的男女们

第二张图中连接两个人的实线和虚线的序列，叫做增广路径（augmenting path），它可以用来增大配对的数量。1957年，克劳德·伯杰证明任何未达到最大可能配对数的匹配方式都对应一个增广路径。敌友国理工学院的计算机科学家写了一个简单的算法来找到这些增广路径，结果让参加实验的98%的人都成功配对。

不久以后，敌友国最高法院裁定，允许同性结婚。于是学院在网站上张贴启事，招募所有性别取向的志愿者。这下关系图更复杂了，出现了许多彼此交叠的三角恋爱关系，如下所示。

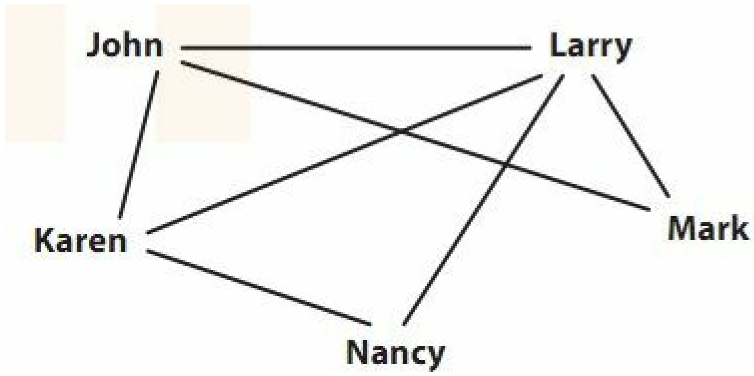


图**3-5**　打破传统的男男女女

这导致寻找增广路径的简单算法不好用了。研究者转而求助于杰克·埃德蒙兹的工作。在1965年，埃德蒙兹写了一篇标题很文艺的论文“路径、树木与花朵”（Path, Trees and Flowers），里面介绍了一种稍为复杂的方法，能够在广义好友关系图中找出增广路径。敌友国理工学院采用埃德蒙兹的思路，从而能够让97%的参与第二个实验的人找到合适的伴侣。

“路径、树木与花朵”中的思想，除了能为解决广义好友关系图中的配对问题提供一种有效的方法，还有更大的影响。埃德蒙兹的算法为100个人找到最佳匹配需要大概100^4个计算步骤。对于今天的计算机，计算100^4（即1亿）次一点也不费劲，而更原始的穷举方法大概需要的计算步骤数是2后边跟78个0。埃德蒙兹在论文里还写了很多题外话，探讨高效算法的本质。他意识到不存在能够完全抓住“效率”直观概念的严格定义，于是他试着提出自己对高效的见解，即存在一个计算方法，它解决这种规模的问题所花费的时间是“算术级别的”，比如100^4、100^2或100^{12}。后来这一类型的问题有了另一个人们更为熟知的名字——P（代表polynomial，多项式级别的，它代替了埃德蒙兹的“算数级别的”），人们用它来指代那些可以高效解决的问题。这就是P/NP问题中P的一面。

3.4　团问题

敌友国理工学院的一个社会学教授想做一个实验，为此需要找到彼此互为朋友的50个居民。她自己办不到，就向几个计算机科学家请教这个问题，他们说手头刚好有哪些人是朋友的数据。“找50个互为朋友的人组成的团是吧，应该没问题。”一个计算机科学教授说。

但是有问题，而且是个很难的问题。从敌友国里挑出50个人的不同方案的个数，是一个151位的天文数字，根本不可能逐个检查。研究者们为此想尽了办法，比如他们意识到，如果一个人拥有朋友的数量不到50个，那么这个人不可能在一个50人的团里。即便如此，他们绞尽脑汁，却连25个互为朋友的人都找不出来，而且也不能有效证明整个敌友国不存在50人的团。

就在他们想放弃的时候，有一个研究生说：“不是有个阿尔法会吗？”阿尔法会（Alpha Society）是一个富有传奇色彩的、半地下社会组织，据说里边人人都是朋友。计算机科学家设法找到了50个阿尔法会会员的名字（幸好这个组织只是“半地下”）。有了这50人的名单，就只要检验1225对人是不是朋友就可以了。结果他们真的每两个人都是朋友，这让计算机科学家们惊讶不已，而阿尔法会会员们表示淡定。50个互为友人组成的团就这样被找到了。

3.5 “递棍儿”

让我们看看一个求解简单的问题，怎样经过一点小小的改动，就变成一个非常难求解的问题。

敌友国的小孩子们玩一种叫“递棍儿”的游戏。他们把一个小棍子传来传去。“递”的意思是在把棍子从一个孩子传到另一个时，两个人一起把它握住。

规则有两条：

1. 只能给你的朋友递棍儿；
2. 每对朋友之间必须递一次棍子，且只能递一次。

假设5个小孩在玩这个游戏。

从Barbara开始，她把棍子递给Eric，Eric递给Alex，然后棍子依次经过Cathy、Eric，最终被递给David，游戏成功结束。

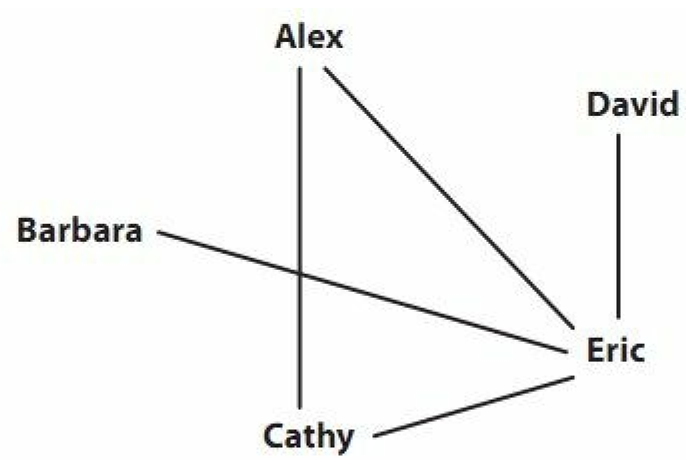


图**3-6**　孩子们

玩了几次孩子们很快发现游戏能成功结束的前提是，拥有朋友个数为奇数的人不能超过两个：如David和Barbara分别有1个朋友，是奇数，而Cathy和Alex分别有2个和4个朋友，是偶数。为什么呢？因为除了第一个和最后一个人，其他人接棍子的次数和把棍子递给朋友的次数必须相

等，所以这些人中间的每个人经手棍子的次数应该是偶数。

如果每个孩子都有偶数个朋友，那么游戏也能玩成，只要保证开始和结束时是同一个人就可以了。

比如孩子们可以这样传棍子：从Alex开始，棍子依次经过Eric、David、Barbara、Eric、Cathy，最终回到Alex手上。

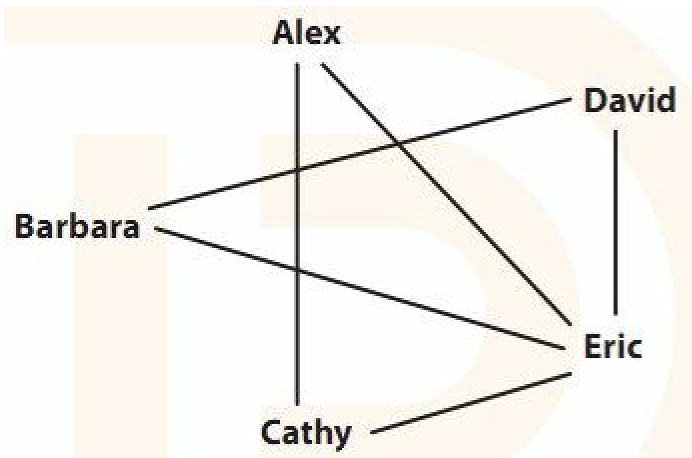


图**3-7**　每个孩子都有偶数个朋友

“递棍儿”的灵感来自19世纪的一个著名难题。在普鲁士的哥尼斯堡（今俄罗斯加里宁格勒）有七座桥横跨于普雷格尔河上，请看这张老地图。



图**3-8**　柯尼斯堡的桥

哥尼斯堡的市民想知道能否有一条路线跨过每座桥一次，并且只跨过一次。1735年，著名的数学家莱昂哈德·欧拉着手解决这个问题，他画了一张图。

和“递棍儿”游戏中的孩子们不同的是，北区和岛区、岛区和南区之间都有多重的“友谊”，即桥梁。然而原理是一样的，欧拉证明，没有人能将每座桥跨过并且只跨过一次，因为四块地区都有奇数条桥梁与其他地区相连。

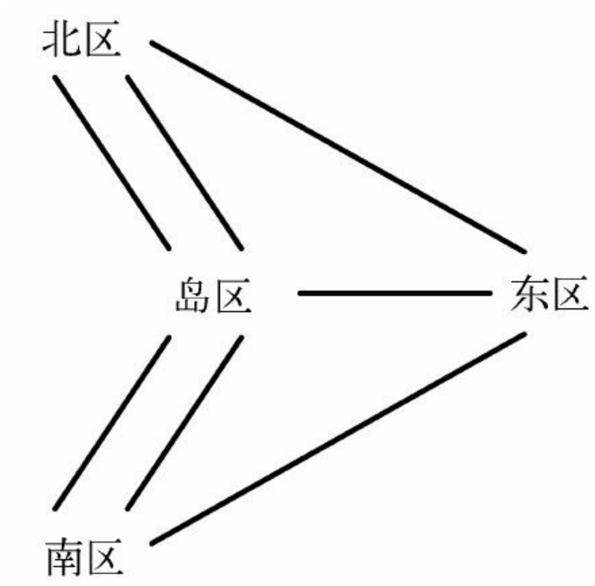


图**3-9**　欧拉的难题

因为源于欧拉和哥尼斯堡七桥难题，“递棍儿”游戏的成功完成方式叫做欧拉回路。玩的人比较多时，完成游戏的方式可能有很多种，但孩子们可以很容易地判断能不能玩成，只要数数有几个人的朋友个数是奇数就行了。

随着敌友国的孩子们渐渐长大，他们觉得“递棍儿”太简单了，没劲。于是他们稍微改了一下规则，新游戏起名为“递棍儿2”（真没创意）。修改后的规则如下：

1. 只能给你的朋友递棍儿；
2. 棍子必须且只能每个人经手一次，第一个递出棍子的人除外，因为棍子最终要还到他手里。

以下边这张朋友关系图为例，David把棍子递给Barbara，然后棍子依次经过Eric、Alex、Cathy，最终递回David手上。

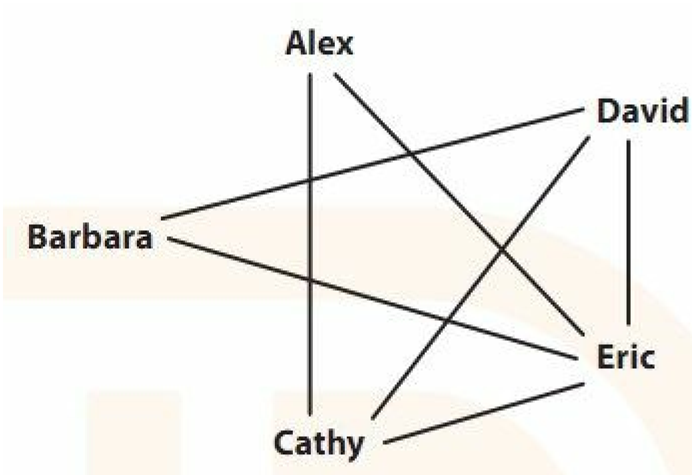


图**3-10**　可以完成的**“**递棍儿**”**

而在接下来这张图里，孩子们发现不可能完成“递棍儿2”。

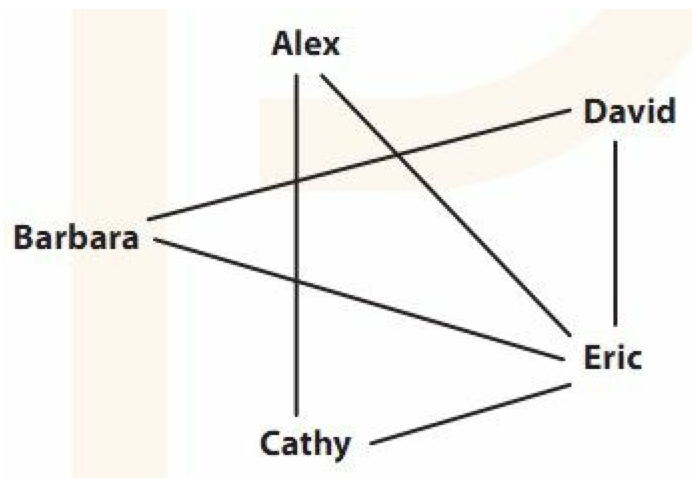


图**3-11**　不可能完成的**“**递棍儿**”**

由于规则变简单了，孩子们以为“递棍儿2”会更容易解决。但是当人数变得较多时，“递棍儿2”的难度陡然上升。1857年，数学家威廉·罗恩·哈密顿发明了一个叫Icosian的数学游戏。我们可以用“递棍儿2”来描述它的玩法，为了让图更简单，我们把每个人用名字的首字母代替。

如果顶点代表敌友国的人，每条边代表一对朋友之间的关系，就会得到

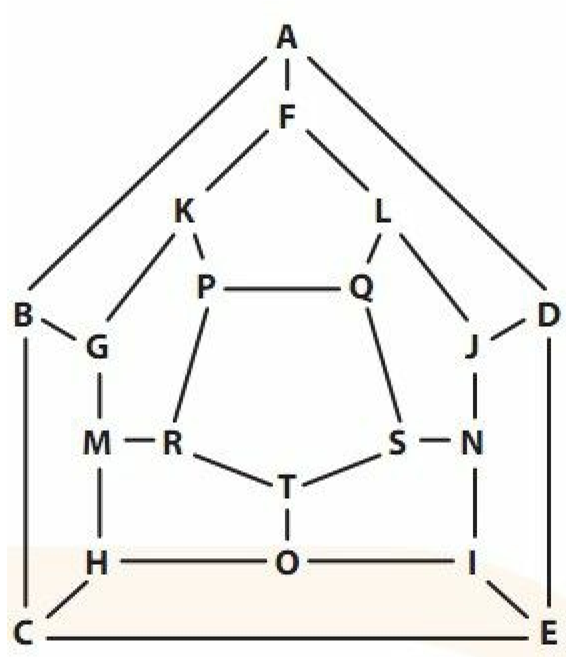


图**3-12　Icosian**难题

Icosian游戏来自正十二面体，即一个有十二个平面的球。

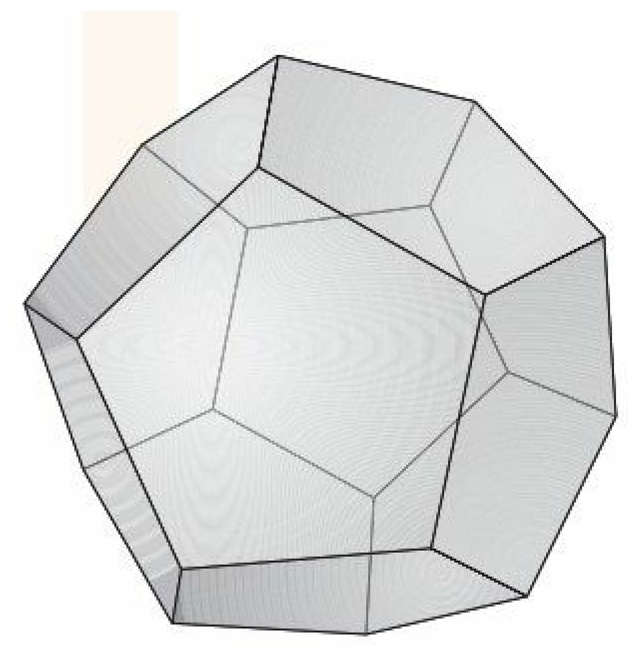


图**3-13**　十二面体

如果顶点代表敌友国的人，每条边代表一对朋友之间的关系，就会得到

上面那张好友关系图。

为了纪念Icosian游戏的发明者，“递棍儿2”游戏的成功完成方式叫做哈密顿回路。

3.6 刷房子

敌友国政府颁布了一个新法案，要求所有公民为了城市的美观，必须把自家的房子刷成和所有邻居的房子不一样的颜色，无论邻里之间是敌是友。很多人表示抗议，说这是强制让民众白花钱、做苦工。后来政府同意用公共资金为居民刷房报销全款，但有一个条件：刷什么颜色要由政府来定。

由于要买的涂料数量巨大，政府职能机构想尽可能减少购买的涂料颜色的种类。少用一种颜色就能省下几百万美元。政府给敌友国理工学院拨了一笔钱，让他们找到最少需要几种颜色的涂料才能保证所有相邻的房子都有不同的颜色。

敌友国每家最多有12个邻居。所以最容易想到和实现的方案需要13种颜色：每个邻居刷一种不同的颜色，然后中间的房子刷第13种颜色。但敌友国的研究者觉得他们还能做得更好。

1852年，南非数学家弗朗西斯·格思里在为英国各郡的地图填色时，猜想是否只用四种颜色，就足够让所有地图上每两个接壤的地区有不同的颜色。当时很多数学家都考虑了格思里提出的问题，接着在1879年和1880年出现了两个四色问题的“证明”，分别由阿尔弗雷德·肯普以及彼得·泰特发表。11年后两个证明都被发现有致命错误，在之后几乎一个世纪的时间，这个问题都未能得到解决。

1976年，肯尼斯·阿佩尔和沃尔夫冈·哈肯终于给出了四色问题的证明，但他们使用了一种非常有争议的技术。证明过程需要用一个计算机穷举所有可能的情况来支持他们的论点。传统数学家不相信这样的证明，因为它不能完全用人的思维来检验。但是几十年过去了，人们没有找到阿佩尔-哈肯证明中有任何错误，如今大多数人都相信，四种颜色足够为所有的地图填色。

四就是极限了吗？三种颜色够不够填满所有地图？不够，来看内华达州和它接壤的州。



图**3-14** 内华达州

加利福尼亚州、俄勒冈州、爱达荷州、犹他州和亚利桑那州，这几个州在内华达州外面围成一圈。由于这个圈包含奇数个州（5个），需要至少三种颜色来填充它们；如果只有两种颜色，比如绿和蓝，加利福尼亚州涂绿色，与之接壤的俄勒冈州必然是蓝色，然后爱达荷州必须是绿色，犹他州是蓝色。现在，亚利桑那州分别与绿色的加利福尼亚州和蓝色的犹他州接壤，所以它既不可以是绿色，也不能是蓝色。所以我们至少需要三种颜色——蓝、绿和黄，来填充这五个州。

内华达州与五州分别接壤，那么它不能填蓝、绿或黄这三种颜色。所以我们需要第四种颜色——红，来为内华达州上色。

有了基于四色定理的证明的高效算法，就可以很快为任意地图涂上四种颜色。于是敌友国的研究者找到了用四种颜色给所有房子刷漆的方法，保证了相邻房子的颜色不同。政府又敦促学院找到只用三种颜色涂料的方法。恰巧敌友国没有邻居个数为奇数的房子，所以研究者没办法直接排除只用三种颜色的方法存在的可能性。

计算机科学家们几经尝试，最终放弃了。他们向政府报告找不到只用三

种颜色的涂色方案。政府不得不使用四种颜色的方案粉刷所有房子。敌友国的计算机科学家们将来在申请政府津贴时会遇到更多困难。

3.7 分组

敌友国小学的老师想把500个学生分成两组。每个人都想和朋友在一起，所以老师想尽可能避免把一对好友分到两个组去。让我们用“递棍儿”游戏中的那张好友关系图来看看。

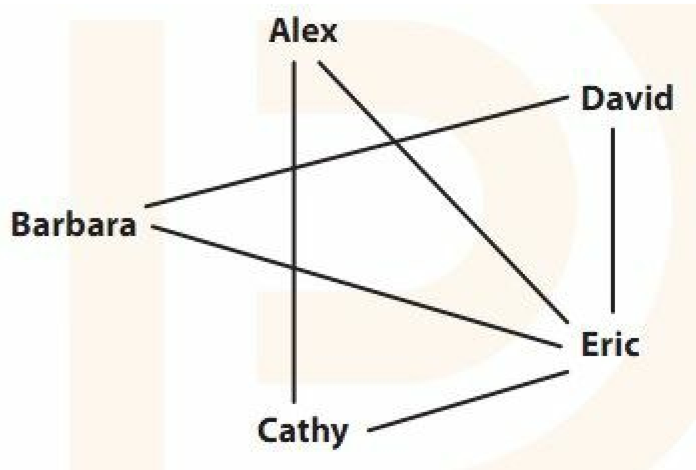


图**3-15**　小学

最好的分组方法是把Alex和Cathy放在一组，Barbara、David和Eric在另一组，这样只拆散了两对好友，Alex—Eric以及Cathy—Eric。不存在只拆散一对好友的方案。

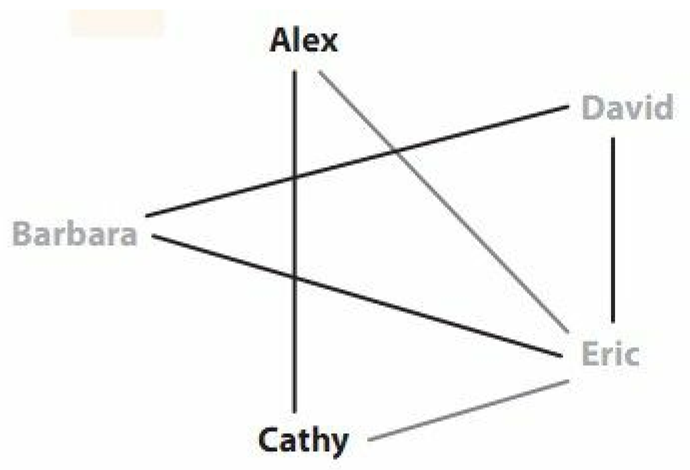


图**3-16**　小学的分组

敌友国小学的校长向学院求助。有很多有效的方法可以解决这个问题，它被称为最小割问题，因为老师想把被分割的好友数减到最小。研究者针对这500个学生给出了一个很好的分组方案，只拆散了17对好友。

从此以后所有人都过上了快乐的生活。说“从此以后”不太准确，事实上只过了一天，老师就发现，把很多互为仇敌的小孩分到一组太糟糕了，比拆散好友更不可取。于是校长回到学院，请研究者找到一种把尽可能多的互为敌人的学生分到不同组里的方案。校长觉得第一次的分组问题解决得那么轻松，这次应该也不会太难。校长想错了。

在这个新问题里，研究者需要尽可能多地把敌人分割开，这叫做最大割问题。但是不像最小割问题，计算机科学家不知道哪种算法能最大化被分割到两个组的仇敌数目。

也不是完全无计可施。学院利用1995年由米歇尔·戈麦斯和大卫·威廉森写的算法，找到了一个将1321对敌人分割到两个组的方案。尽管没能找出最好的方案，但他们知道不存在把1505对及以上的敌人分割到两组的方案。校长有点失望，他以为学院能找到最好的方案，但现有方案也还算合理。从此以后所有人都过上了更加快乐或更加郁闷的生活。

3.8　P和NP

让我们看看哪些问题让敌友国的研究者们头疼不已：团、哈密顿回路（“递棍儿2”）、地图填色，以及最大割。这些问题有一个共性，都可以用很简单的方法检查方案的有效性。一旦敌友国的科学家找到了阿尔法会的成员，就很容易验证是不是所有成员之间都互为朋友，即阿尔法会成员组成的是一个团。孩子们如果听说了一个能完成“递棍儿”游戏的方法，只要顺着方法玩一遍就知道对不对。随便给出一个涂色方案，政府很容易检查它是否符合相邻的房子颜色不同。在最后一个例子里，对于任意一个分组方式，校长只要数数有几对敌人被分开就能判断方案的好坏。

这一类能够很快验证一个解的有效性的问题，计算机科学家给它们起了一个名字：NP。如果你非想知道的话，这两个字母代表nondeterministic polynomial time（不确定性多项式时间）。团、哈密顿回路、地图填色和最大割都是NP中的典型例子。

与之形成对比的是P类问题，即我们能很快找到最优解的这些问题：最短路径、配对、欧拉回路（“递棍儿1”），以及最小割。

也许存在能快速找到团的高明算法。可能将来某个聪明的研究生能想出如何轻松找出哈密顿回路，或是一个最大化分割敌人到两个小组的快速步骤。也许团、哈密顿回路、地图填色和最大割也是在P类问题中的，就像最短路径、配对、欧拉回路和最小割一样。也许NP中的每个问题都有一个快速的解法，我们只是不知道而已。

这就是P/NP问题。P=NP意味着美好世界的到来，每个NP中的问题都对应一个有效的解法，也就是说每个有解的问题都能很快地验证给定解，并且快速地找到其中最优的解。与此相反，如果哪怕有一个NP中的问题我们找不到能给出快速解法的算法，那么P还是不等于NP。

判断P是否等于NP是计算机科学乃至整个数学领域最关键的问题。许多人呕心沥血，试图找到解决团、哈密顿回路或其他问题的好算法，可惜都失败了。与此相反，要证明P≠NP，必须证明无论现在还是将来，都不可能存在能够快速找到团或解决其他NP问题的算法。如何证明一件

事不可能做成？目前这两个研究方向都没有什么实质性进展。

正是认识到了P/NP问题的重要性，克雷数学研究所才决定为解决它的人提供一百万美元的奖金。也正因为它无比关键，才促使我写出了这本书。

3.9 敌友国之外

这章给出的只是几个例子，除此之外还有成千上万个我们不能快速求解的NP问题。要是你觉得除了计算机科学家和子虚乌有的敌友国居民之外，没有人会对P/NP问题感兴趣，那么请让我列举一小部分来自其他学科领域的例子，我们同样不知道下面这些问题的有效算法。

1. 生物学

人类有23对染色体。每条染色体上都排布着由碱基对组成的基因序列，碱基对有四种，即腺嘌呤（A，adenine）、胞嘧啶（C，cytosine）、鸟嘌呤（G，guanine）和胸腺嘧啶（T，thymine）。染色体序列可能以ACTGATTACA开始，可能会很长。最短的序列有4700万碱基对，最长的有2亿4700万碱基对。当代DNA测序技术一次只能测定大约20～1000个碱基对。生物学家需要想办法把许多短小的序列片段拼接成完整的序列。拼接序列是一个很难的计算问题，也是一个NP问题，因为如果我们提前知道了要拼接的DNA序列的全貌，我们可以用相对较快的方式检查某种拼接方式是否与之匹配。由于找不到解决这个NP问题的有效算法，生物学家需要更多的序列才能绘制出基因组图谱。而且很可能绘制出的图谱上有更多的空白和错误。研发出剪接序列的最优算法将显著提升基因图谱的质量。

DNA序列能编码信使RNA（mRNA）的序列，后者负责蛋白质的表达，而蛋白质则在组成生物细胞的几乎所有功能中发挥着关键性作用。蛋白质执行何种功能通常依赖于特定的几何形状，而mRNA序列决定蛋白质如何折叠，从而具有这种形状。蛋白质折叠的计算过程仍然是生物学一大未解之谜，研究P/NP问题对理解蛋白质折叠有着重要的启发作用，进而帮助人类预防和抵抗疾病。

蛋白质折叠识别（protein threading）是一种根据mRNA序列来预测蛋白质折叠方式的统计学方法。即使是这种对于理解蛋白质折叠有很大局限性的方法，也需要解决困难的NP问题。

2. 物理学

寻找物理系统能量的最低值是一个NP问题，物理系统可以是相互作用的磁性粒子群，或是一堆肥皂泡，等等。人们不知道如何有效找到这些系统最低的能量状态。难道物理系统不是总会最终达到最低能量状态吗？不一定。

考虑一个弹珠放置在如图3-17所示的曲面上。这个弹珠最终停在3.0处才能达到最低势能。如果它一开始处于1.0位置，它将保持静止，除非推它一把。从中可以看出，物理学系统并非总会最终达到最低能量状态。找到复杂系统的最小能量，对于计算机和系统本身来说，都很困难。

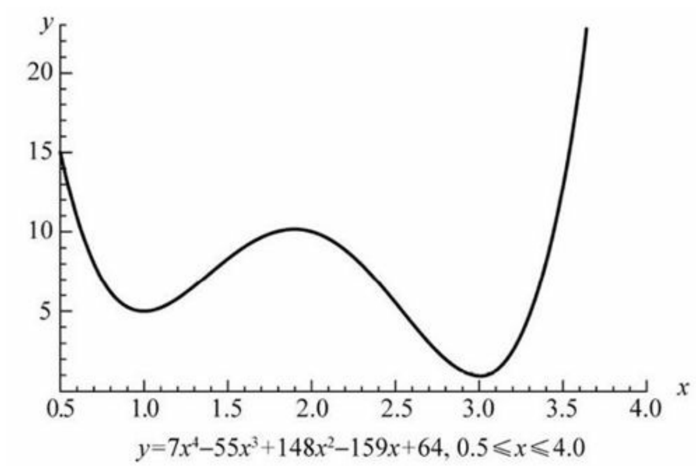


图**3-17**　物理系统的图表

量子力学可能帮我们解决部分困难的NP问题，而不是全部。这部分内容我们将在第9章深入讨论。

3. 经济学

一个对冲基金经理需要从许多复杂的投资工具中作出选择。一个预算有限的消费者走进超市，面对各种商品难以取舍。他们其实都在求解困难的NP计算问题。由于人们并不是总能有效解决这些问题，通常做出的决策都是次优的。市场上的这些计算效率低下对于我们的经济乃至整个社会的不利影响究竟有多大？这是个很棒的问题，可惜我们无法给出很

棒的答案。

约翰·纳什是一个经济学家，传记图书《美丽心灵》（*A Beautiful Mind*）和同名电影都从一定程度上展现了他的生平。纳什获得（迟到的）诺贝尔奖，是因为他证明了个体间根据策略互动时存在一个均衡状态，在此状态下所有个体采取的策略经过博弈达到均衡，即任何一方改变策略的做法，都不会为任何人带来利益。纳什的存在性证明没有给出如何找到这些策略的方法，计算机科学家目前得到的证据显示，找到这样的策略可能是一个计算上困难的问题。问题的难度预示着市场本身并非总能找到这样的均衡状态，也就是说市场将保持持续波动，人们也将不断改变策略以追求更好的结果。

4. 数学

1928年，数学泰斗大卫·希尔伯特提出了他伟大的判定性问题（Entscheidungsproblem），即能否找到一个能正确判定任意数学命题真伪的算法。1931年库尔特·哥德尔证明，必然存在某些命题，它们根本不能被任何公理系统的集合证实或证伪。受哥德尔工作的影响，几年后，阿隆索·丘奇和阿兰·图灵分别独立证明了不存在满足判定性问题要求的算法。

但如果我们做出限定，即命题所需证明的长度必须短到可以在一本很薄的书上写出来，这样有可能找到判定算法吗？我们可以通过计算来求解这个问题，只要穷举对于给定数学命题的所有可能的短小证明就可以了。这是一个NP问题，因为如果给定一个证明，我们容易判断它是否有效，但是找到一个有效证明实际上很困难。这也是为什么数学家能通过巧妙地证明某些数学命题而博得声誉的原因。

有时候找出让某些形式简单的逻辑表达式为真的方法都很困难。对这个问题的研究直接促成了一个能将大部分NP问题都联系起来的理论的形成。我们在下一章讲讲这个故事。

3.10 Icosian游戏的一个解

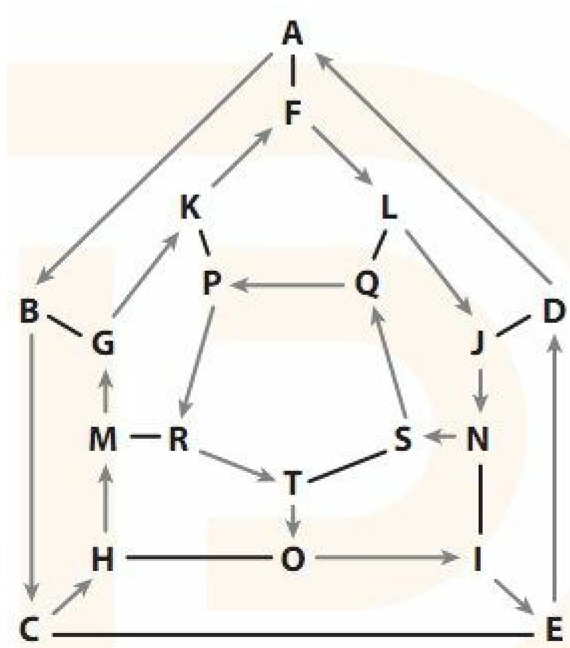


图3-18 Icosian的解

第4章　NP中最难的问题

> 心理学家以数学家为研究对象做了一个实验。把数学家关在一个小木屋里，地上放一些引火物、一张桌子，桌子上有一桶水。然后心理学家点着了地上的引火物。数学家提起桌上的水桶把火扑灭了。
>
> 到目前为止一切正常。心理学家再次进行实验，还是把数学家关在那个有桌子、水桶和引火物的小屋里，但这次，水桶是放在地上的，靠近那堆引火物。然后心理学家又放了火。数学家提起水桶，把它放到桌子上，然后就等着。心理学家和同事们好容易才把数学家及时从即将烧塌的小木屋里救出来。
>
> 心理学家问数学家："为什么不像上次一样把火扑灭？"数学家回答："我已把问题归约到一个之前解决过的情形。"
>
> ——一个老生常谈的数学笑话

4.1　第一个NP完全问题

汤姆·赫尔在1970年担任多伦多大学计算机科学系的主任，他想聘用斯蒂芬·库克，当时加州大学伯克利分校刚刚拒绝了库克的终身教授职位申请。库克喜欢帆板运动，于是赫尔带他到安大略湖玩，给他展示在多伦多附近也能进行帆板运动，和旧金山湾是一样的。这个小伎俩奏效了，斯蒂芬·库克于1970年秋天加入了多伦多大学的教席。当年的赫尔真可谓慧眼识珠，因为在那之后不久，库克就成为加拿大最著名的计算机科学家。

库克研究逻辑学和计算机科学的联系。那年秋天，他向即将在来年五月召开的第三届ACM计算理论年会（ACM Symposium on the Theory of Computing，STOC）提交了一篇论文。这篇论文最初的结论已经被遗忘了，不过它在当时引起了足够大的兴趣，被安排在会上宣读。在准备会议期间，库克重写了论文以加入最新的工作成果，而这篇名为“理论证明过程的复杂度”（The Complexitiy of Theorem-Proving Procedures）的论文，将首次把P/NP问题呈现在世人面前，进而改变历史。

为更好理解库克的论文，让我们回头看看上一章的团问题。团是一群互为朋友的敌友国居民。在下面这张好友关系图中，Alex、Cathy和Eric是一个团，而Alex、David和Eric则不是，因为Alex和David是敌人而非朋友。

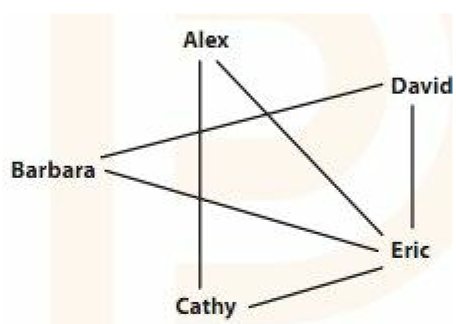


图**4-1**　团

你可能记得，在敌友国有一个半地下社会组织叫阿尔法会，据说里面的成员都互为朋友，也就是说他们构成一个很大的团。假设阿尔法会真的

是一个团，我们能否从上面的好友关系图看出一些端倪？首先不能排除任何一个人是阿尔法会成员的可能。Alex可能在阿尔法会，David也有可能。但是Alex和David不可能同时在阿尔法会，因为他们是敌人。所以要么Alex不在阿尔法会，要么David不在。让我们把这个逻辑表达式写下来。

```
Alex not in Alpha Society OR David not in Alpha Society
```

这里的`OR`不具有排斥性，即有可能Alex和David都不在阿尔法会。由于Alex和Barbara是敌人，所以我们也知道这两个人不可能同时在会里：

```
Alex not in Alpha Society OR Barbara not in Alpha Society
```

这两个逻辑表达式必须同时为真，所以我们有：

```
(Alex not in Alpha Society OR David not in Alpha Society) AND
(Alex not in Alpha Society OR Barbara not in Alpha Society)
```

既然Barbara和David是朋友，他们可能都在阿尔法会，然而这个逻辑与上述表达式并不冲突。把完整的关系图用逻辑表示出来，我们得到：

```
(Alex not in Alpha Society OR David not in Alpha Society) AND
(Alex not in Alpha Society OR Barbara not in Alpha Society) AND
(David not in Alpha Society OR Cathy not in Alpha Society) AND
(Cathy not in Alpha Society OR Barbara not in Alpha Society)
```

如果Alex、Cathy和Eric在阿尔法会，而Barbara和David不在，表达式为真，因为每个`OR`项中都因存在不是会员的人而得到满足。然而，如果Alex、David和Eric在会中，则表达式为假：`(Alex not in Alpha Society OR David not in Alpha Society)`这部分不满足，因为Alex和David都在会中。

表达式很好地表达了团问题。当且仅当阿尔法会的成员们组成一个团时，表达式为真。

我们可以为敌友国的2万居民生成一个类似的逻辑表达式，将它简写为Φ。Φ可能很长，有几百万个字符，但是计算机能很容易地存下它。如果阿尔法会真是一个团，则表达式为真。

我们有另一个表达式Ψ50，它在阿尔法会有至少50个成员时为真。我不想具体讲如何写出Ψ50，它的构造方法和最早的电子计算机做加法的方式是一样的。

如果把两个表达式结合起来，我们就得到(Ψ50 AND Φ)，它在“阿尔法会是一个团，且至少有50个成员”时为真。反过来说，如果阿尔法会令(Ψ50 AND Φ)为真，那么它是一个至少有50个元素的团。

假设我们有一个快速算法，它能告诉我们给定的逻辑表达式是否能被满足。如果我们给它(Ψ50 AND Φ)，然后它回答：“是的，(Ψ50 AND Φ)可以被满足。”则敌友国存在一个50人的团。如果(Ψ50 AND Φ)不可以被满足，则那里不存在这样的团。解决了这个可满足性问题，你就能解决团问题。

我们刚刚描述了一个计算机科学中最关键的概念：归约。我们把寻找团的问题归约到检查逻辑表达式的可满足性的问题。现在只要我们有解决可满足性问题的算法，就可以通过很容易的变换，得到团问题的求解算法。团问题的求解至少和可满足性的求解一样容易。如果可满足性是容易求解的，那么团问题也是。如果团问题没有有效解法，那么可满足性问题也没有。

能归约到可满足性问题的不仅是团问题，而且还包括我们前面讨论过的其他NP问题，包括旅行推销员问题、哈密顿回路、最大割和地图填色问题。事实上，斯蒂芬·库克证明了NP中的每一个问题都能通过某种方式归约到可满足性问题。解决了可满足性问题，你就能解决所有的NP问题。如果你有解决可满足性的有效算法，我们就能快速解决所有容易检查特定解有效性的问题，也就是说证明了P=NP。只要你有一个解决可满足性的有效算法，那么克雷研究所提供的几百万美元奖金就是你的了。

1971年5月4日，在当时的俄亥俄州榭柯高地斯托弗的萨默塞特酒店召开的STOC上，斯蒂芬·库克宣读了自己的论文。

> 结果表明，可满足性问题可以作为一个相对有用但尚未存在有效计算方法的待选课题来研究，并且我认为值得投入更多精力来证明这个猜想。找到它的证明将可能带来重大的突破。

P/NP问题从此诞生了。

4.2　21个问题

库克的论文刚刚发表时，尽管结果的重要性得到了肯定，但也没有立即改变学术研究的方向。毕竟对可满足性问题感兴趣的人不多，并且大家对NP问题还不熟悉。斯蒂芬·库克甚至没有在论文中提到NP这个缩写，而用了它的全名“不确定性多项式时间”（nondeterministic polynomial time）。但不久以后，情况随着伯克利教授理查德·卡普的一篇承上启下的论文而改变了。

看过库克的论文以后，卡普意识到有一种方法可以把可满足性归约到团问题。卡普建立了一个简单的计算过程，它能把给定逻辑表达式转换成相应的好友关系图，并有当且仅当关系图上存在等于某个数的团时，表达式为真。任何能解决团问题的算法都可以用来解决满足性问题。库克表明了可满足性是NP问题中最难的。而卡普表明，团问题至少和可满足性问题一样难。这样一来，团问题就成为NP最难的问题之一。和可满足性问题一样，如果我们找到了团问题的快速解法，那么所有NP类问题就迎刃而解，从而证明P=NP。

库克表明了如何把类似团这样的问题归约到可满足性问题。卡普则表明可满足性可归约为团问题。这两个表面看上去非常不同的问题，在计算角度上竟然是相同的。可满足性容易计算的充分必要条件是团问题容易计算，而后者的充分必要条件是P=NP。

卡普不仅证明了团问题是NP中最难的问题之一，而且找出了其他19个同样难度的重要问题，包括分割难题、旅行推销员、哈密顿回路、地图填色以及最大割。只要有效解决了这些问题中的任意一个，你就能解决所有问题并证明P=NP。如果说P≠NP，那么这21个问题（包括可满足性问题）没有一个能被快速地解决。

卡普并不是凭空创造了这21个问题，对解决这些问题感兴趣的也不仅是数学家和计算机科学家。其中的许多问题来自现实世界。

比如可口可乐公司，它旗下的饮料品牌超过3000个，行销世界各地。一个可口可乐的灌装工厂可能会生产几百种不同的饮料产品。每个工厂都有数台机器需要进行一系列的作业，混合每种饮料所需的部分原料。这

些机器可以生产各种产品，而灌装工厂需要合理调度机器的作业，以达到其吞吐量的最大值，从而在最短时间内生产出最多的饮料产品。这是一个作业调度问题，它也在卡普列出的21个问题之中，其求解难度不亚于任何其他的NP问题。

从电子计算机诞生之日起，科学家和程序员就在尽最大努力，研发解决作业调度问题的最佳算法，因为一个很小的调度改进就可能为某个企业节省数百万美元。还没有人写出对所有调度问题都能给出最好方案的算法。卡普证明，甚至连调度的简单改进也是一个难度不亚于任意NP问题的难题，这也立即揭示了为什么人们找不到那些调度改进算法。

不仅是大公司会关心这些问题。假设你带家人去迪士尼乐园春游，因为放假所以要排长队。你想尽可能多玩些好的娱乐景点，减少排队浪费的时间。《迪士尼乐园非官方指南》的作者想出了一整套如何减少等待时间的游玩攻略，但同时他们也承认自己在做一件很困难的事。

> 魔法王国景区21个景点一日游，针对成人的可能游览方案总数是让人震惊的51 090 942 171 709 440 000种。那可是超过5100亿亿种的组合，大概是地球上沙粒总数的6倍。要是再考虑上快速排号系统（FASTPASS）、游行、吃饭和休息等，就更复杂了，组合的数量还得增加好多。

类似的问题也困扰了科学家们很多年。 比如，快递公司会关心如何安排每个送货员的路线，让行驶距离最短，从而节省时间和燃料。 事实上，这类“如何以最小的代价访问多个地点”的问题十分普遍，人们给它们起了一个外号：旅行推销员问题。

许多科学家尝试过找出旅行推销员问题的最佳解法。还有人试过为作业调度问题找到一个高明的算法。另外人们还尝试过求解团、最大割以及其他卡普所列出的算法。卡普的论文意味着所有这些研究者研究的其实是一个问题，因为如果某人为其中某一个算法找出了有效的解法，那么所有其余的算法就迎刃而解了。

同理，所有这些独立工作的研究者之中没有一个人能对他们所研究的特定问题找到有效的解法，这很可能暗示了P≠NP，或者至少可以说，很难为这些问题中的任意一个找到好的算法。所以作业调度的技术员们可以理直气壮地告诉老板，我们是找不到机器上最好的作业调度方法，但

不只是我们不行，连奥兰多的那帮机灵鬼都搞不定迪士尼乐园的游览线路规划呢。

而卡普的工作，一下子就把所有这些出了名的难以解决的计算问题联系到了一起。从此，P/NP问题成为众人瞩目的焦点。

每年计算机协会都会颁发ACM图灵奖，其地位相当于计算机科学界的诺贝尔奖。该奖得名于数学家阿兰·图灵，他在20世纪30年代为计算机科学奠定了基础。ACM将1982年的图灵奖授予斯蒂芬·库克，以表彰他为P/NP问题的建立所做出的杰出贡献。但一个图灵奖还不够，理查德·卡普于1985年也获得了图灵奖，奖励他在算法理论方面的杰出工作，特别是列出了21个NP完全问题。

4.3 起个好名字有那么重要吗

卡普的论文给出了我们今天使用的P和NP这些名字。但是该怎么称呼那些NP中最难的问题呢？库克用了一个非常学术的名称deg({DNF 重言式})，而卡普使用了术语（多项式时间）完全。可是这些名字太别扭了。

高德纳接手了这个命名问题。鉴于他一贯杰出的研究工作以及宛如丰碑般的三卷巨著《计算机程序设计艺术》[1]，高德纳在1974年获得图灵奖。由于认识到了P/NP问题至关重要的作用，高德纳想在第4卷中最终敲定这个NP中最难问题集合的命名。1973年，他通过普通邮件做了一次投票调查。他直到今天都坚持不用电子邮件，这在业界是很有名的。但在1973年，只有普通邮件可用。

1 《计算机程序设计艺术》卷1 、卷2、卷3、卷4A的英文版均已由人民邮电出版社出版，中文版也会相继呈现给读者，详情请登录图灵社区（www.ituring.com.cn）查阅相关图书。——编者注

高德纳在投票中给出的几个候选项——herculean（艰巨）、formidable（可怕）和arduous（费劲），都不怎么受欢迎。他收到了许多来信建议，有些很直白，如intractable（顽固）和obstinate（执拗），有些则更加口语化，如hard-boiled（原意是煮得很硬的蛋，可能是在向库克致敬[2]）和hard-ass（抗揍的混蛋，简直像可满足性问题一样难啃）。

2 库克（Cook）这个姓的原意为“厨师”。——译者注

最终获胜的来信建议是“NP-complete”（NP完全），它由新泽西州贝尔实验室的几个人提出，是那里的研究者经过多次讨论之后产生的。“complete”（完全或完备）这个词出自数学逻辑学，一个事实集合被称为完备的，就是说它能解释某些逻辑系统的所有真命题。类似地，“NP完全”表示这些NP问题的集合强大到能够用来解决任何其他的NP问题。

高德纳对这个民选结果不太满意，但也没有觉得它差到让人活不下去的

地步。他本人特别想要找一个英文词，既能捕捉“困难的搜索问题”这个直观的意象，又要琅琅上口，便于向大众普及。1974年，高德纳总结了他的调查过程，文中写道：“‘NP完全’这个名字其实学术气息有点儿浓，不好向大众推广，但也没有差到不能用的地步。”

“NP完全”很快成为标准术语。高德纳用了大概40年才完成巨著的第4卷。

其实高德纳当年本应该坚持己见，为“NP完全”，甚至“P”和“NP”这些概念敲定一些不那么学术的术语。P/NP问题的重要性已经远远超出了计算机科学领域，而这种将术语的首字母缩写的命名方式，阻碍了门外汉对其重要性的认知。但这些术语在几十年后已经在文化中根深蒂固，即使现在我们想到了更好的名字，恐怕也不好纠正了。

高德纳也意识到，如果最终证明P=NP，也就是说NP完全问题其实就是P中的问题，那么之前为它起名字而做出的种种努力岂不全都白费了。但是高德纳说了：“即使将来可能觉得尴尬，我也愿意冒这个险——我将奖给第一个证明P=NP的人一只活蹦乱跳的火鸡。”所以现在证明P=NP的人可以赢得100万美元，外加一只火鸡。

4.4 超越卡普的工作

在卡普的论文之后，计算机科学界掀起了一个寻找并证明各种问题是否属于NP完全的热潮。在接下来的几年里，教授和研究生们成功证明了许多已知的搜索问题（以及一些新问题）是NP完全的。一本1979年出版的书[1]中列举了三百多个主要的NP完全问题。NP完全问题持续在各个领域涌现，如计算机科学、物理学、生物学、经济学，以及其他许多攀登到困难顶峰的学科。Google学术搜索NP-Complete将返回超过138 000篇科研文献，时间跨度从1972年到2011年，单是2011年就有近1万篇。我们在此无法一一列举，只从中挑选几篇，看看这些文章是什么风格。

1 *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 作者Michael Garey和David Johnson（New York: W. H. Freeman, 1979）。

1. 支配集（**Dominating Set**）

敌友国是否存在这样的50个人：其余的人和这50个人中的至少1个是朋友？NP完全问题。

2. 三角切分问题 （**Partition into Triangles**）

敌友国理工学院的每个宿舍只能容纳三名学生。我们能把所有敌人都安排在不同的宿舍里吗？NP完全问题。

3. 大规模数独游戏

数独（Sudoku）是一种起源于日本的填数字游戏，使用一个9×9的格子，如下图所示。

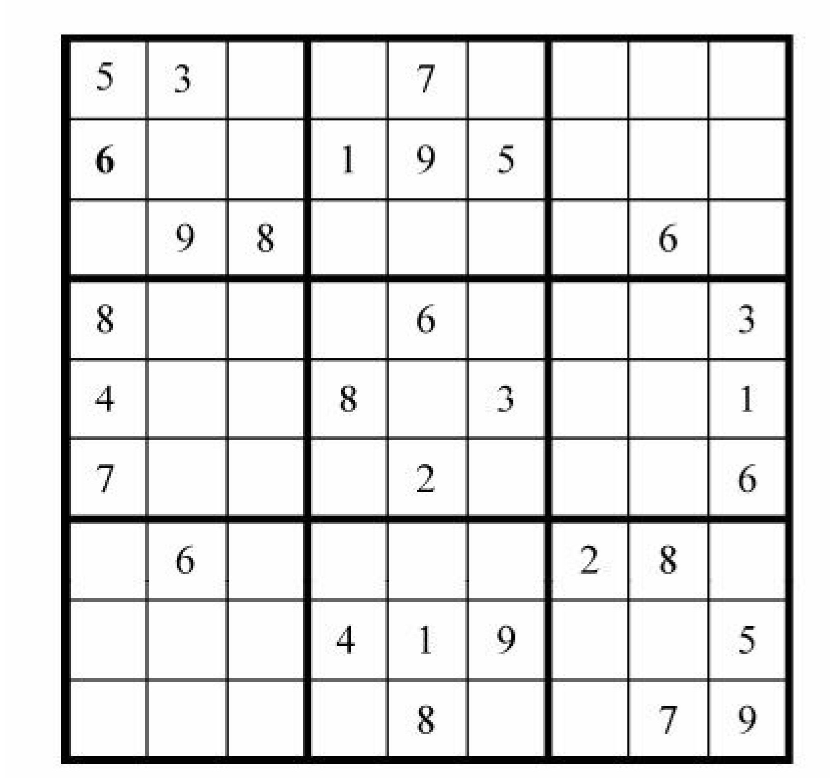


图**4-2**　数独

数独的游戏目标是填满所有空格，并使每一行、每一列和由黑线标出的3×3小方格中的数字分别是1到9的不重复的数字。

5	3	**4**	**6**	7	**8**	**9**	**1**	**2**
6	**7**	**2**	1	9	5	**3**	**4**	**8**
1	9	8	**3**	**4**	**2**	**5**	6	**7**
8	**5**	**9**	**7**	6	**1**	**4**	**2**	3
4	**2**	**6**	8	**5**	3	**7**	**9**	1
7	**1**	**3**	**9**	2	**4**	**8**	**5**	6
9	6	**1**	**5**	**3**	**7**	2	8	**4**
2	**8**	**7**	4	1	9	**6**	**3**	5
3	**4**	**5**	**2**	8	**6**	**1**	7	9

图**4-3**　数独的解

数独是NP问题，因为很好检查一个解。找到一个解有多难？不太困难。使用简单的回溯算法，普通计算机可以在几秒内找到一个有效解。

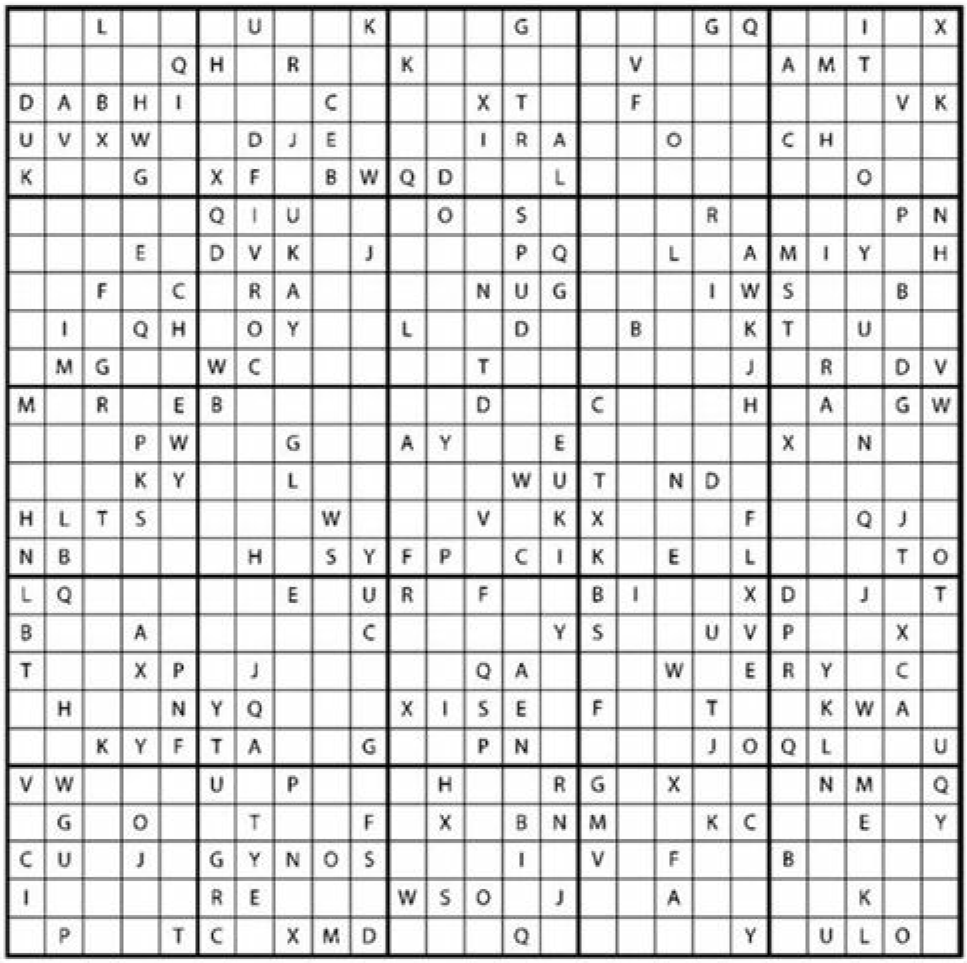


图**4-4**　大规模的数独

但是如果谜题的规模变得更大，像上面这个25×25的版本，要求每行、每列和小方块中填入A到Y且不重复的字母，会发生什么呢？

这回普通计算机就得算好长时间了，而100×100的数独游戏能打败当今最快的机器。

大规模的数独游戏是NP完全问题。你自认为是数独高手？如果你能可靠地解决大规模的数独，那么你也能解决可满足性问题、旅行推销员问题，还有其他几千个NP完全问题。

数独可不是桌面游戏中唯一一个NP完全问题。来看微软Windows系统自带的扫雷游戏。



图**4-5**　扫雷

每个小方块埋着一个数字或者地雷，数字代表临近的方块（包括水平、垂直和对角线相邻）总共有几个地雷。在某个不是地雷的方块上点一下会显示出下面的数字。如果你觉得方块下面有地雷，就标一个小旗子。点中地雷你就输了。判断能否玩赢一个大规模的扫雷游戏也是NP完全问题。这是上面那张图里剩下的地雷。


MineSweeper - Google Chro...
www.chezpoor.com/minesweeper/minecore.htm
Game Options Help

图**4-6**　扫雷失败

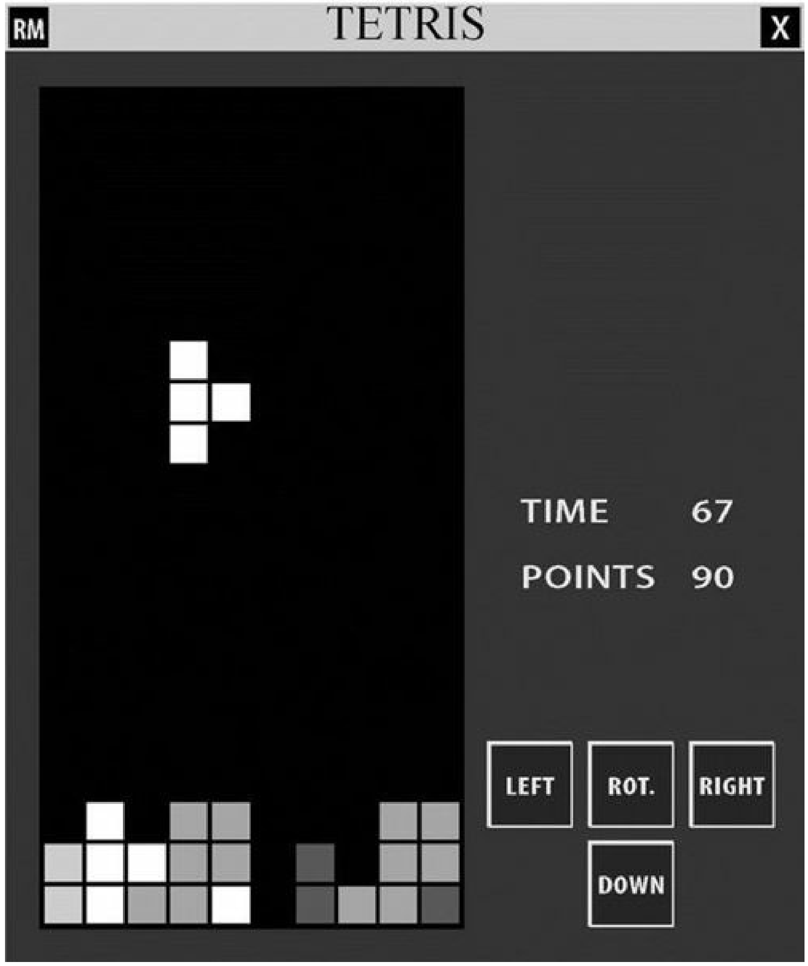


图**4-7**　俄罗斯方块

当然还有俄罗斯方块。玩家平移和旋转各种下落的积木，填满一行的那一刻整行会消失，游戏目标是不让整个画面都被方块填满，坚持尽可能长的时间。

积木分很多种形状。

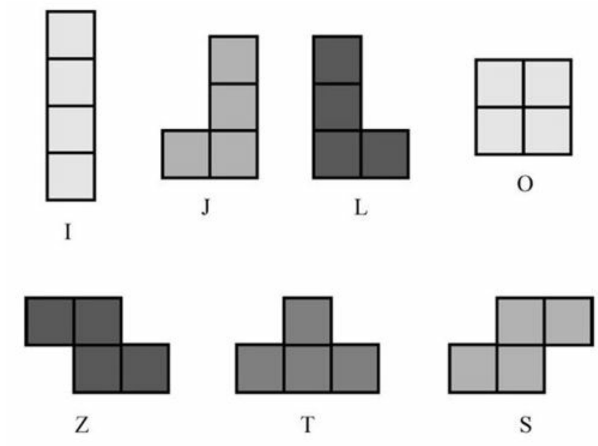


图**4-8**　俄罗斯方块的积木

经典俄罗斯方块游戏中你不知道下一块积木的形状。但即使你提前知道了各种形状的积木到来的次序，如何把俄罗斯方块玩到最好也是一个NP完全问题。

谁曾想把数独、扫雷和俄罗斯方块这些游戏玩好，就可以证明P=NP从而解决我们这一代面临的最大的挑战呢？



图**4-9**　魔方。图片作者为**Tom van der Zanden**

魔方又怎么样呢？即使是3×3×3的小魔方，一般人也得花不少工夫才能知道如何复原，而求解更大的魔方的难度可想而知。

其实不算太难。我们对于解决更大规模的魔方复原问题有一些很快的算

法，它们都基于一个叫做群论的数学分支。这些算法不能保证找到全局最少步骤的解法，但是总能找到足够短的方法，从任何可能的打乱方式开始将魔方复原。

与玩好俄罗斯方块、扫雷和数独的困难程度相比，复原魔方的问题容易得出奇。

而双人对战游戏，如国际象棋、跳棋、黑白棋和围棋，它们的难度又如何呢？这些游戏的大规模版本，其难度和可满足性问题等NP完全问题的难度是相当的。但是这些双人游戏不属于NP类问题。比如我告诉你，这盘国际象棋肯定是白方赢，最后一步是三列卒吃王，你根本没法验证我的话。计算机科学家基本上都认为国际象棋、跳棋、黑白棋和围棋的难度均大于任何NP问题。

4. 肾脏交换

正常人有两个健康的肾来将体内的代谢废物滤出体外。如果只剩一个肾，人还是可以过正常的生活。如果两个肾都不能用，就必须用透析来维持生命，这不仅需要耗费大量金钱和时间，而且时刻都有可能死亡。

有两个健康肾的人可以捐献出其中一个，挽救双肾衰竭的病人，前提是捐献者的肾和受捐者的身体相容。通过简单的抽血就可进行这种相容性测试。

假如Alice不幸双肾衰竭，她的丈夫Bob同意捐献自己的一个肾。如果Bob的肾和Alice的身体相容，那么可以通过手术将Bob的一个肾摘除并移植给Alice。

但Bob的肾有可能受到排斥。还有一线希望，就是所谓肾脏交换项目。

假如另一个人Charlie也需要肾，愿意为他捐肾的是他兄弟David，而David的肾也和Charlie不相容。然而，如果David的肾与Alice相容，并且Bob的肾和Charlie的肾相容，那么就可以安排让四人同时上手术台，这样术后每个人都有一个好肾，挽救两条生命。

假设我们有了关于上述这样潜在的肾脏捐献者-接受者的数据库，就可

通过有效的算法，让尽可能多的两对捐献者-接受者成功配对，达成肾脏交换。基本上这就是前一章介绍过的配对问题，很容易解决。

不止是两个人，还可以有更多人来配对。2011年年末，60场手术为30位患者移植了健康的肾脏，这是通过其他任何方法都不可能做到的。

然而如果允许多于两对的捐献者-接受者互换肾脏，如何找到尽可能多的多人搭配方案在目前还是一个NP完全问题。P=NP的证明在这里是性命攸关的，可比如何玩好扫雷重要得多。

4.5　漏网之鱼

人们在20世纪70年代所研究的大部分NP问题，都要么被认定属于NP完全问题，要么被有效解决，从而归属于P。但还是有一些NP问题冥顽不灵，难以确定其归属。这些问题中的一部分在很多年后被解决了，另一部分直到今天都还没有被定性，成为漏网之鱼。

1. 图的同构问题

在敌友国流行着一个叫“刀剑征程”（Blade Quest）的MMORPG（大型多人在线角色扮演游戏）。游戏中每个玩家都会起一个新名字，操纵一个虚拟角色（avatar）去和其他玩家的虚拟角色进行互动。敌友国的敌友关系也被引入了刀剑征程的世界。在现实中互为朋友的居民，他们在游戏中的虚拟角色也是朋友关系；相反，现实中的敌人在游戏里也是敌人。

Isabel、John、Kevin、Laura、Molly和Nancy是几个喜欢玩刀剑征程的敌友国居民。

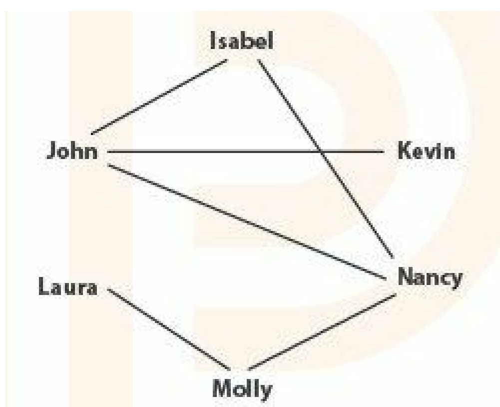


图**4-10**　玩刀剑征程的居民

这6个人在游戏中的虚拟角色的名字是Achris、Bolem、Chyrard、Degarald、Enthrr和Fev，但是哪个虚拟角色对应哪个真人是保密的。在刀剑征程中，这几个虚拟角色也有一个好友关系图。

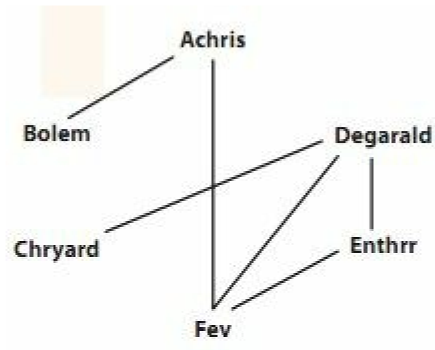


图**4-11**　刀剑征程中的虚拟角色

Laura看了这两张图，就在游戏里给别的玩家发了一个消息："我知道你在现实中是谁。"你能猜出来吗？

这两张图存在唯一的对应方式，即Isabelle在游戏中对应的虚拟角色是Enthrr，John是Degarald，Kevin是Chryard，Laura是Bolem，Molly是Achris，还有Nancy是Fev。我们可以验证一下，如Molly在敌友国的朋友是Laura和Nancy，与之对应，在刀剑征程中Achris的朋友是Bolem和Fev。

好友关系图之间的映射被称为图的同构问题。有可能存在多种映射，也可能不存在，这取决于给定的图集。图的同构性属于NP类问题：如果知道了谁对应谁，你可以通过逐对查看好友关系在两张图中是否一致来验证这个解的有效性。

但是图同构是否属于P，即我们是否能找到图之间的映射（假设存在）的有效算法，还是未知的。我们也不确定图同构是NP完全的，而出于多种技术原因，计算机科学家也不认为它是NP完全的。图同构属于另一小撮问题，其难度比P问题大，但比不上哈密顿回路、最大割等NP完全问题。

2. 质数和因数分解

你可以对15进行因数分解，得出15×1或5×3。24这个数可等于24×1、12×2 、8×3或6×4。但是你只能把17写作17×1。17是一个质数，即只能把它写作1和自身乘积的数。最开始的几个质数是2、3、5、7、11、

13、17、19。质数有无限多个。已知的最大质数（在我写作之时）是一个12 978 189位的数，它开始的几位是316 470 269 330 255 923 143 453 723…

怎么判断一个数是否是质数？例如要判断1 123 467 619是否是质数，你需要遍历所有小于1 123 467 619的数，看看是否有能整除它的数。事实上你只需要检查到33 518（1 123 467 619的平方根）就可以了。听起来是挺快，可是你如何判断下面这个数是否为质数？

8 273 820 869 309 776 799 973 961 823 304 474 636 656 020 157 784 206 028 082 108 433 763 964 611 304 313 040 029 095 633 352 319 623

早在古希腊就出现了判定质数的算法，但直到20世纪70年代才出现对几百位数有效的算法。这些算法基于数学中一个叫数论的领域的研究成果，在检查时需要使用随机生成的数。虽然这些算法在实践中表现很好，但是不依赖于随机生成数的质数判定算法直到2002年，才由一位印度教授曼尼达·阿加拉瓦尔带领他的学生尼拉吉·卡亚尔和尼汀· 萨克斯纳发现，从而把该问题归属于P类问题。

这些算法虽然能告诉我们

8 273 820 869 309 776 799 973 961 823 304 474 636 656 020 157 784 206 028 082 108 433 763 964 611 304 313 040 029 095 633 352 319 623

这个数不是质数，但令人惊奇的是，它们不能告诉你这个数的因子有哪些，也就是说不能对其因数分解。

我们有判定质数的有效算法，但是没有分解因数的有效算法。

因数分解是一个NP问题，因为如果你看到了
84 578 657 802 148 566 360 817 045 728 307 604 764 590 139 606 051
和
97 823 979 291 139 750 018 446 800 724 120 182 692 777 022 032 973
这两个数，把它们相乘就可验证乘积等于上面的
8 273 820 869 309 776 799 973 961 823 304 474 636 656 020 157 784 206 028 082 108 433 763 964 611 304 313 040 029 095 633 352 319 623

计算机科学家认为因数分解不属于P类问题，也觉得它不是NP完全问题。它是很难，但不像可满足性问题或地图填色问题那样难。

不只是喜欢摆弄数字的数学家们认为质数判定和因数分解问题很重要。现代密码学的许多方法都依赖于某些难以分解因数的大数，这是第8章的话题。

3. 线性规划

敌友国有一个熟食店叫Frenemy Fancy Franks，里面出售四种口味的香肠：法兰克福香肠、意大利香肠、德国香肠和西班牙香肠。不同种类的香肠制作时间不同，售价也有差别。Frenemy Fancy Franks应该如何安排每种香肠制作的数量，以获取最大的利润呢？

找到最优的方案就是在满足各种限制条件的前提下使收入最大化。假设制作一根法兰克福香肠的成本是1美元，意大利香肠的成本是2美元，德国香肠是3美元，西班牙香肠是4美元，且每日的总预算是1万美元。那么，1乘以法兰克福香肠的数量，加上2乘以意大利香肠的数量，加上3乘以德国香肠的数量，加上4乘以西班牙香肠的数量，结果应不超过1万。

在这类条件下进行最优化选择的问题叫做线性规划。可能的解集在高维空间形成一个几何体，叫做多面体。

早在1947年，格奥尔格·丹齐克发明了单纯形法（simplex method），这种方法能很快解决线性规划问题。单纯形法遍历多面体所有的边，直到找出最佳的解集。

既然有了单纯形法，我为什么要把线性规划放在（20世纪70年代）未分类的问题里面呢？因为在某些罕见的个例中，单纯形法无法快速求解线性规划问题。

1979年，利奥尼德·卡奇安发明了椭球算法（ellipsoid algorithm），基本思路是把多面体切分成越来越小的块，不断缩小范围并最终找到最优解。卡奇安给出了椭球算法有效性的证明，从而把线性规划分到了P类问题中。然而实践中椭球算法的耗时比单纯形法要长很多。虽然缺乏实

用性，但椭球算法在之后的几十年中启发了许多更复杂的算法的诞生，因而它在学术上还是很有影响力的。

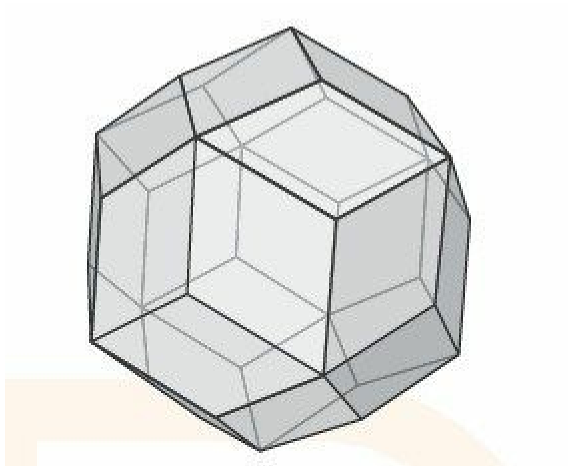


图**4-12**　多面体

这样一来，线性规划在理论和实践上分别都有了好的解决算法——只不过是两种非常不同的算法。

1984年，纳伦德拉·卡马卡发明了内点算法（interior point algorithm）。和单纯形法类似，卡马卡的算法遍历多面体，只不过是在内部遍历而不是表面。和椭球算法一样，内点算法也能将线性规划归属到P类，并且在经过优化后该算法的实际性能甚至高于单纯形法。

以上就是解决线性规划问题的三种非常不同的算法。第一种（单纯形）在实践上表现很好。第二种（椭球）在理论上很好。而第三种（内点）则在理论和实践上的表现都很好。能对一个在20世纪70年代末还被认为是未解决的问题给出这三份答案，是很值得人们为之自豪的进步。

第5章　P和NP诞生前的历史

> 不要怕蛮力搜索，而是要合理使用它。[1]

1 Georgy M. Adelson-Velsky Vladimir Arlazarov and Mikhail Donskoy *Algorithms for Games (Programirovanie Igr)* (New York: Springer-Verlag 1988).

在上一章中我们讲到了高德纳最终没能为NP完全问题找到一个更恰当的英语单词。如果他当初把眼光投向东方的俄罗斯，就会发现一个合适的词perebor（Перебор），意为“使用蛮力来搜索”，即尝试所有可能性以找到最佳方案。P/NP问题其实是在问，我们是否必须用蛮力搜索来解决团问题，还是存在更快的方法也能达到目的。

但是当年高德纳和其他美国人一样，都很难了解俄罗斯发生的事情。从1945年第二次世界大战结束起，一道铁幕将当时的苏联与东欧，以及美国与西欧隔绝开来。从20世纪50年代开始，冷战导致美国和苏联两国围绕着科技发展展开了一系列竞争，两国都想赢得这场知识领域的“军备竞赛”。由此带来的一个负面影响是基本断绝了东西方科学界的互访和交流。尽管情况在20世纪70年代有所缓和，但直到1991年冷战结束，东西方才恢复了全面的科研交流和讨论。如今，随着大部分的学术成果都被放到网上供人查阅，再加上学者可以较为自由地到世界各地进行学术访问，我们可以认为全球同属一个学术研究社群，而不是被人为地分成两个互不来往的社群。

这一章讲述了两个故事，两条通往P/NP问题的道路。结局是西方的斯蒂芬·库克和东方的列昂尼德·莱文分别率先提出了是否P=NP的问题。但是科学并不是凭空发生的，两边的科研都经过了漫长的路程，最终才可能产生库克和莱文的工作。在本章，我们会截取这两段历史的几个片段进行讲述，包括西方对于高效能计算的本质的苦苦探索，以及东方对蛮力搜索的必要性的理解。最终我们会发现二者殊途同归，走到了P/NP问题面前。

5.1　西方

对高效能算法的探索可追溯到大约3000年前，人类第一次使用基本算术来做大数的加法。而我们的故事从20世纪30年代开始，当时人们开始建立算法过程的理论。

1.阿兰·图灵

我们为探索太空而发明了望远镜，借助它我们能看到宇宙遥远的地方，了解它早期的历史。我们建造了能看到原子的显微镜，也建造了巨大的机器用以击碎微小的粒子从而找到更小的粒子。我们甚至破译了人类的DNA。但是有一样事物，它出现在我们的桌上、车内，甚至口袋里，但我们仍然觉得它很神秘，这就是计算机。计算机到底是什么？

computer（计算机）这个词早在17世纪就出现了，那时还没有人想象过机器可以做数学运算。那时的computer是计算员，即做计算工作的人。刚刚兴起的银行业需要计算员来记录和追踪每一笔存款和借贷的账目。

根据《牛津英语字典》，第一次将computer用于指代能计算的设备是在1897年，而电子计算机投入使用是在20世纪40年代。computer的意义，从一种人变为几台庞大的机器，又变成了人们的日用品。

不要把计算机看成是机器，想想它的功能：接收信息，根据指令处理信息，然后产生输出。从功能角度讲，邮政服务是一台计算机：接收信件，解码地址，然后把信件分发投递到合适的位置。生物学过程也是一台计算机：接收DNA序列，产出维持生命所需各种功能的蛋白质。

而我们称之为计算的过程又是什么呢？有没有不能被计算的事物？这个谜题在电子计算机问世之前的1936年，就被伟大的数学家阿兰·图灵解开了。图灵研究了数学家的思考方式，并模拟这种思维过程发明了一个形式化的数学模型，我们现在称之为图灵机，已成为计算学的标准模型。

阿兰·图灵1912年生于伦敦。20世纪30年代初，他考入剑桥大学国王学

院，展露出过人的数学才华。那段时间里，他从一个数学家的角度，思考了计算的本质。数学家在思考时有一定的方法步骤，尽管人的记忆容量非常有限，但是记录信息的纸和笔则可以敞开供应。数学家会在一页纸上匆匆写下一些笔记，写完一张纸后要么再用一张纸，要么回到之前写过的某张纸，对笔记进行一些修改。图灵受这个过程的启发，创造了一个形式化的计算模型，即我们所熟悉的图灵机。

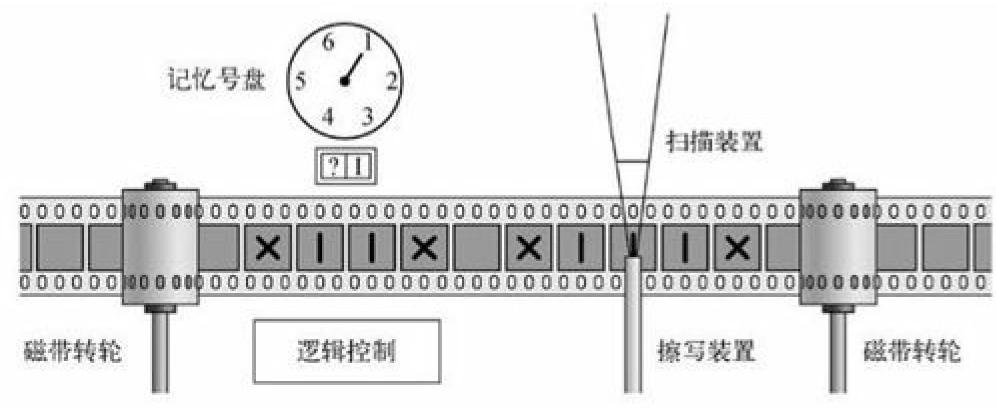


图**5-1**　图灵机

尽管结构非常简单，但图灵宣称，他的机器模型可计算一切能被计算的事物。几乎在同一时间，阿隆佐·丘奇也宣称它的Lambda Calculus（λ演算，一种原始的编程语言）能做同样的事情。丘奇–图灵论题虽然在现代计算机发明之前就已产生，但它通过了时间的考验。图灵机能计算现有或将来一切能被计算的事物。任何其他计算机器模型的能力都不会超出图灵机。

你不一定非要理解图灵机的工作原理才能理解计算。只需要想想任意一种编程语言，假设可利用的存储空间是无限的。所有编程语言在功能上是等效的，并且其计算能力都等于简单的图灵机的计算能力。

图灵在1936年就指出，图灵机并不是什么都能计算。最著名的例子是停机问题，即没有计算机能通过查看一段代码就知道自己是会永远执行下去还是会最终停止。

在第二次世界大战期间，图灵在英国的密码破译工作中承担了重要的任务。战后，他开始考虑图灵机是否是以人脑为原型的。他引入了今天我们称之为“图灵测试”的判定方法，判定机器是否可以表现出类人的智

能。假设你通过即时通信软件和别人聊天，你能判断和你聊天的是真人还是计算机程序吗？如果一个计算机能让大部分人误以为它是真人，它就通过了图灵测试。

不幸的是图灵的学术生涯半路夭折。他于1952年因同性恋行为被当时的英国司法机关判定有罪，这最终导致了他在1954年自杀。直到2009年，英国首相戈登·布朗才为当年政府对图灵的判决发表了一份官方道歉声明。

美国计算机协会以图灵的名字命名了计算机界的最高荣誉：图灵奖（等同于其他领域的诺贝尔奖），以表彰图灵为计算机科学和人工智能所做的开创性工作。本章提到的许多科学家都是图灵奖得主。

2.计算复杂度

电子计算机从20世纪50年代开始在社会上发挥重要作用，人们需要某种方法来衡量要解决的不同问题的计算难度。第一批方法的建立源于科学家对人类思考和交流方式的描述。

1943年，心理学家沃伦·麦卡洛克和沃尔特·皮茨建立了一个描述脑活动的模型，名叫神经网（neural net）。20世纪50年代，逻辑学家斯蒂芬·克莱尼建立了图灵机的一个受限版本，叫有限自动机（finite automaton），并研究了能被其解决的问题的属性。有限自动机对于描述简单机器（如卖汽水的机器）的控制逻辑非常有用，但过多的限制使它无法表示更复杂的算法。

20世纪50年代，语言学家诺姆·乔姆斯基试图理解人类如何生成英语或其他语言的句子。他引入了如“上下文无关文法”（context-free grammar）等方法，可生成句子的解析树，如下图所示。

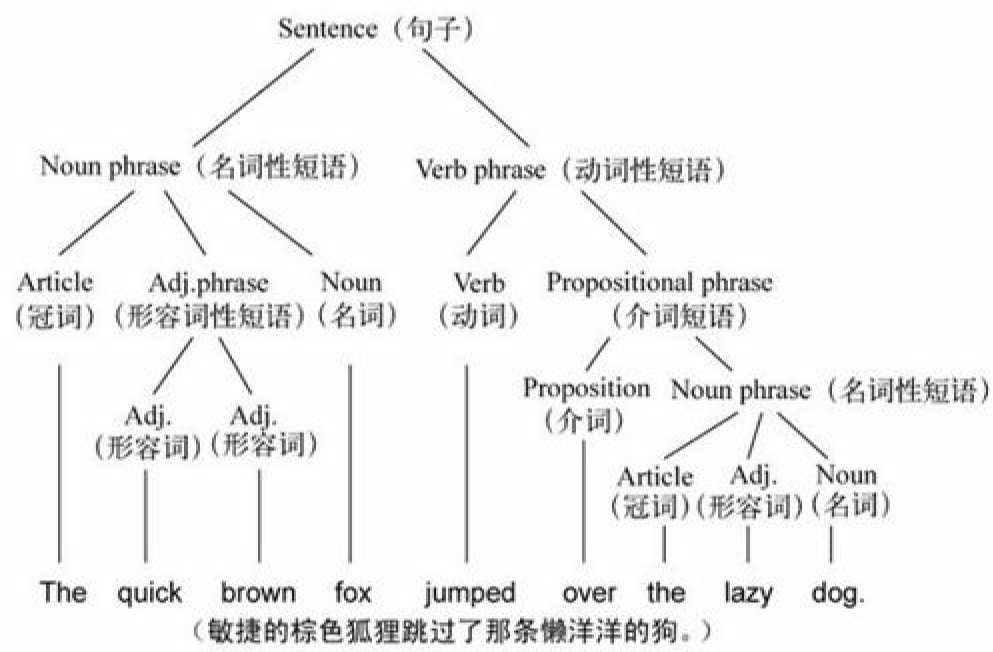


图**5-2**　解析树

上下文无关文法对人类语言句子的表述有一定合理成分，但不够完整。目前语言学界对哪种语法更适于表述语言——抑或这种语法究竟存在与否还存在争议。

而在描述和解析编程语言及XML等计算机生成的数据文件等方面，上下文无关文法十分好用。尽管如此，它和有限自动机一样，都不能表述我们对高效能计算的直观概念。

人们还引入了很多其他的模型，但这个领域的突破性进展发生在1962年，由纽约州斯克内克塔迪通用电气公司科研实验室的尤里斯·哈特马尼斯和理查德·斯特恩斯取得。他们想出一种绝妙的方式，来表示计算机程序解决问题的能力：看看随着问题描述的规模的扩大，计算机算法所需的运行时间和存储空间发生了多大的变化就可以了。两人在1965年发表的论文《论算法的计算复杂度》（On the Computational Complexity of Algorithms）推动了计算复杂度这个研究领域的诞生。因为这些工作，哈特马尼斯和斯特恩斯获得了1993年的图灵奖。

20世纪60年代，计算复杂度领域出现了两个不同的研究方向。其中一个是研究在给定的时间、存储空间和显式计算模型下，哪些问题能够被解决，而哪些不能。另一个研究方向的带头人是曼纽尔·布卢姆，他在麻

省理工学院攻读博士学位期间的工作采取了一种非常抽象的方法，其研究结果不仅不依赖于任何特定的模型，甚至不依赖于特定种类的资源，如时间或存储空间。这两种方法都没有抓住高效能计算的真正本质。

3.P和NP

高效能计算的正确定义来自20世纪60年代的两篇论文，分别是杰克·埃德蒙兹的“路径、树木与花朵”（Paths, Trees and Flowers）和艾伦·科巴姆的“函数的内在计算难度”（The Intrinsic Computational Difficulty of Functions）。

埃德蒙兹论文最重要的贡献是给出了配对问题的首个有效算法，这在第3章讨论过了。论文有一节叫“题外话”，埃德蒙兹在里面讨论了指数和代数级别的区别，不过他不建议为算法的效率设定任何严格的标准：

> 需要稍微解释一下用到的“高效能算法”这个词……我没有准备好给出严格的理论体系赋予其正式的含义，况且这篇文章的上下文也不适合这样做……实践中，区分代数级别和指数级别的重要性，要远大于区分可计算与不可计算的重要性……如果硬要制定一些严格的标准，可能会在实践上阻碍有用的算法的研发，因为这些算法要么是未知的，要么理论上与标准不符……无论如何，为鼓励人们去搜寻优秀而实用的算法，很重要的一点是认识到：质疑此种标准的存在价值在数学上是有一定道理的。

埃德蒙兹的“代数级别”正是P所代表的我们能有效计算的问题。正如埃德蒙兹建议的，虽然有必要用正式的方法来定义“是否P=NP”等问题，但我们心中也应该对高效能计算有一种非正式的概念，而这也是我想在本书中表达的一种观点。

科巴姆同样独立地定义了P，并讨论了为什么这个概念很有用。

有很多原因让类型P成为一个自然而然的概念。首先，如果我们对各种通用类型的计算机器分别做一个形式化的表述，会发现总是离不开几个相同的定义明确的函数。据此，我们能给出一个数学的表征P，并相信它能正确地表征那些定义得不那么正式的类别。

和可计算性的定义一样，类型P并不依赖于计算模型的特定细节。

科巴姆也提醒人们注意：

> 这个问题让人想起如何形式化地定义“效率”这个概念。虽然二者很明显有关联，但是强调的方面不同，这里主要关注的是计算过程的物理方面。

科巴姆也许认识到将来可能出现不属于他所说的类型P的计算模型。随机化和量子计算模型的研发表明，我们也许无法对高效能计算有一个固定的概念。

之后在1971年，斯蒂芬·库克发表的论文定义了NP（即我们能很快验证的问题）、P/NP问题，以及第一个NP完全问题。一年后理查德·卡普在论文中给出了一长串属于NP完全的问题。

1972年，IBM的T. J. 华生研究中心召开了一次有关计算机的计算复杂度的座谈会，这次会议主要因为卡普的论文而被世人记住。组织者在会议结束时安排了小组讨论，探讨这个领域的未来发展。其中一个问题是：“如何把原来零散的下确界结论以及某些算法整合起来，发展成一个更加统一的理论？”对于这个问题的答案，包括卡普本人在内的小组成员都没太有把握，只是模糊地感觉到它可能与库克和卡普的论文有关，与P、NP、可归约性以及之后的NP完全性概念等有关。

卡普意识到他们需要为这个新领域起个好名字：

> “计算复杂度”这个名字太宽泛了，它包括了布卢姆和其他人的工作，而现在我们还没法把他们的工作涵盖进来；“混凝土计算复杂度”听起来好像土木工程的分支[1]；“计算机计算的复杂度”听起来又不是很准确。

1 concrete除了“实际”还有“混凝土”的意思。——译者注

后来这个领域就叫做计算复杂度。P/NP问题的重要性很快显现，抢尽了这个领域其他研究方向的风头。抽象复杂度给人的感觉更不靠谱了。连曼纽尔·布卢姆本人也改变了研究方向，开始转攻密码学和程序检验。布卢姆因其在20世纪60、70和80年代所做的广泛研究而获得1995年

的图灵奖。很多年后，当被问及他在20世纪70年代改变研究方向的原因时，布卢姆简单地答道："库克是对的。"

5.2 东方

当年苏联的“理论控制论”学界活跃着很多重要的人物，而我们的故事将集中介绍其中的三位，他们分别代表处理蛮力搜索的三种方式。

1. 谢尔盖·雅布隆斯基，他首次将蛮力搜索的研究聚焦于寻找实现特定计算函数的最小电路，但他的权力欲和傲慢阻碍了俄罗斯的计算复杂度科学的发展。
2. 安德烈·柯尔莫哥洛夫，他是伟大的俄罗斯数学家，主张通过代数信息来衡量复杂度。
3. 柯尔莫哥洛夫的学生列昂尼德·莱文，他独自研究发现了P/NP问题和NP完全问题，但是由于政治原因，他甚至无法在祖国得到博士学位。

1.谢尔盖·雅布隆斯基

俄罗斯的计算学研究被称为理论控制论，该学科直到20世纪50年代电子计算机在军事上开始发挥重要作用时才正式起步。谢尔盖·弗谢沃洛多维奇·雅布隆斯基生于1924年的莫斯科，二战时曾在苏联军队服役，战后到莫斯科国立大学学习数学。他于1953年获得博士学位，导师是俄罗斯可计算性领域的先驱彼得·诺维科夫。雅布隆斯基和诺维科夫的另一个学生阿列克谢·李雅普诺夫组成研究小组，在莫斯科国立大学举办了一系列关于逻辑函数的研讨会。他们的小组成了俄罗斯计算理论的研究中心。

第4章提到的可满足性问题的处理对象是由基本逻辑运算符AND、OR和NOT组合成的逻辑表达式。我们甚至可以用一串AND、OR和NOT来表达计算过程。有些问题只需要少量的逻辑运算符组成的“电路”，而其他问题需要大规模的电路才能运算。谢尔盖·雅布隆斯基在20世纪50年代研究了这个概念，我们今天称之为电路复杂性。

信息论创始人，美国人克劳德·香农证明某些逻辑函数对应复杂度非常高的电路。雅布隆斯基则研究了生成这些函数的实际难度。听起来是个很困难的任务，但实际上P≠NP的一个强化版本就意味着对于某些描述

起来很简单的搜索问题，不存在小规模的电路可以计算它们。

雅布隆斯基首先指出，根据香农的结论，随机生成的函数对应的电路的复杂度接近最大值。接下来他研究了如何不用随机过程就能找到这样的函数。

需要蛮力搜索全部函数吗？他证明任何能生成具有接近最大复杂度函数的过程，必须内嵌所有函数。那么从某种意义上讲，任何能生成高难度函数的过程都能经过修改，以生成任意其他的函数。雅布隆斯基提出，这意味着蛮力搜索是必要的，因为在生成高难度函数的过程中，必须生成每一个可能的函数。1959年雅布隆斯基将他的论文命名为“论在解决电路理论的某些问题时不可能排除蛮力搜索”（On the Impossibility of Eliminating Perebor in Solving Some Problems of Circuit Theory）。

雅布隆斯基虽然证明了一个重要的结论，但他的解释容易误导人。不能根据“从一个高难度函数能生成任意函数”，就推出“生成高难度函数必须生成所有函数”。雅布隆斯基的结论对于寻找高难度函数的计算复杂度没有什么实际作用。雅布隆斯基的学生佐拉夫勒夫在1960年发表的一篇论文标题同样令人印象深刻：“论不可能通过某种类型的算法构造布尔函数的最小析取范式”（On the Impossibility of Constructing Minimal Disjunctive Normal Forms for Boolean Functions by Algorithms of a Certain Class）。这篇论文同样没有谈到代数运算复杂度的问题。这些研究工作实际上和P/NP问题的研究没有什么关系。

当年的苏联不仅在经济上集权，在学术活动上也是如此。雅布隆斯基断定他自己的研究圆满解决了蛮力搜索问题，所以他强力压制在这个方向上的进一步研究，特别是对计算复杂度和算法的研究。雅布隆斯基后来进入了苏联协调和控制数学研究的数学委员会，成为其中颇有影响力的成员，也在20世纪60年代造成了不少的争端，后面将会讲到。

2. 安德烈·柯尔莫哥洛夫

安德烈·柯尔莫哥洛夫于1903年出生在俄罗斯坦波夫市。1920年他考入莫斯科国立大学，最开始学的不是数学，而是历史学。他研究的问题是：俄罗斯在中世纪的征税工作是以户为单位还是以村为单位来开展

的。他分析了税收数据，证明如果以村为单位来收税，那么对数据的描述会更简单一些。他把这些结果呈交到历史系，得到了极大的赞扬。于是他询问是否能发表论文，得到的回答却是：“你只有一个证据。在历史期刊上发表文章至少需要两个证据来支持你的假说。”于是柯尔莫哥洛夫放弃历史，转而研究那些一个证据就足够的领域。柯尔莫哥洛夫选择了数学，并通过努力，成为20世纪苏联乃至全世界最伟大的数学家，他对几乎所有的数学领域都作出了基础性的贡献。



图**5-3　DILBERT © 2001 Scott Adams**。使用本图片需要得到**UNIVERSALUCLICK**的许可。版权所有，侵权必究

柯尔莫哥洛夫对理解概率和随机性的热忱，直接促成了一个简单却伟大得不可思议的想法。比如下面几串数字：

- 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
- 70710678118654752440084436210484903928483593768474
- 982 922 216 207 267 591 232 795 977 564 268 549 473 337 889 037 097

中有一串是通过随机数生成器生成的，其他两串通过其他方法生成。上面每个数字序列发生的可能性都是相同的，为什么要认为一个序列比其他序列更加“随机”呢？在继续往下读之前，请大家先猜猜哪串数字真的是随机生成的。

很难认为一个全是“9”的序列是随机生成的。第二个序列可能有人知道它是0.5的平方根小数点后数字的前50位。所以第三个数才是随机选择的。

柯尔莫哥洛夫认识到，一个序列的随机性大小可以通过“最少需要多少字数来描述它”来衡量。第一个序列的描述是“50个9”。第二个序列是“1/√2”。而对于第三个数，最短的描述只能是982 922 216 207 267 591 232 795 977 564 268 549 473 337 889 037 097。“描述”是一个不正规的概念，柯尔莫哥洛夫通过计算机程序的表述把它正规化了。

类似的想法在稍早也先后被两个美国人提出过，他们分别是雷·索洛莫诺夫（出生于克利夫兰州，厌恶自己的俄罗斯姓氏）和格里高利·蔡廷。柯尔莫哥洛夫及其弟子则深化了这个概念，所以这个衡量标准被称为“柯尔莫哥洛夫复杂度”。

随机序列就是指最短描述是本身的序列，比如

982 922 216 207 267 591 232 795 977 564 268 549 473 337 889 037 097

通过随机选择每一位数很容易就能生成随机序列。没有随机数就没有这些随机序列。不存在非随机化的算法能生成任意长度的随机序列。我甚至不知道

982 922 216 207 267 591 232 795 977 564 268 549 473 337 889 037 097

是否真的是一个随机序列，因为必须要测试所有比它短的描述方式，而有些表述方式可能非常复杂。

柯尔莫哥洛夫复杂度的理论背景深厚，并且在机器学习、算法分析和计算复杂度等多个领域有广泛应用。虽然它本身没有直接涉及P/NP问题，但是通过对它的研究，柯尔莫哥洛夫的学生列昂尼德·莱文直接导出了P/NP问题。

3. 列昂尼德·莱文

1961年，李雅普诺夫从莫斯科国立大学转到新西伯利亚国立大学任教，在那里他建立了理论控制学系。新西伯利亚市距离莫斯科有2800公里之遥，是俄罗斯第三大城市，也是西伯利亚最大的城市。

系里的老师大部分是雅布隆斯基和李雅普诺夫以前的学生，来自莫斯

科。新组建的理论控制学系很快成为俄罗斯理论控制学的第二大研究中心。鲍里斯·特拉腾布罗在40岁就已经是这个领域的资深研究者，他在中心建成之时就已加入，并很快成为领军人物。1962年，Y. M. 巴兹丁从里加市拉脱维亚大学获得博士学位，并加入了新西伯利亚的研究中心。特拉腾布罗开始和巴兹丁合作，研究计算复杂度的基础理论，许多拉脱维亚和新西伯利亚的学生慕名而来。在20世纪60年代，他们开始建立一个复杂度的算法理论，类似于同时期西方的研究。

但苏联的计算复杂度理论的发展并不顺利，正如特拉腾布罗1984年写的那样：

> 我们与“主流”的控制论研究者们（主要是雅布隆斯基）之间的关系恶化了，对此感到担心。他们对于将算法理论引入复杂度研究领域的态度十分消极……他们不相信计算复杂度和算法复杂度在蛮力搜索领域起到的作用。这些学术上的分歧很可能因为蛮力搜索上的争议而加深，特别是当雅布隆斯基在那样一个协调和控制数学研究的组织中担任高位之后。[1]

1 * B. A. Trakhtenbort, “A Survey of Russian Approaches to Perebor Algorithms,” *Annals of the History of Computing 6*, no. 4 (October 1984).

柯尔莫哥洛夫于1963年夏天访问了新西伯利亚，他和特拉腾布罗分享了学术成果，探讨了算法理论如何促进对信息和复杂度的理解。

在那次访问之后，柯尔莫哥洛夫很快到了基辅大学，并访问了一个为数学和物理学神童专设的高中寄宿学校。他随便问了几个问题，很多问题都被一个叫列昂尼德·莱文的15岁孩子解答了。柯尔莫哥洛夫后来邀请莱文到莫斯科大学跟他学习。莱文在研究生期间的杰出工作分为两个方向。

首先，莱文发明了一个通用的搜索算法。假设Alice对Bob说她有能快速求解团问题的算法，但是不告诉Bob是什么算法。莱文的技巧能让Bob创造出一个几乎和Alice的算法一样快的算法，而他不需要知道Alice的算法具体是什么。莱文通过柯尔莫哥洛夫复杂度的一个变种发展了自己的理论，基本上是尝试了所有合理的方式，才取得了这个成果。

对搜索的研究使莱文转而思考那些能反映搜索本质的问题，于是他引入

了“通用搜索问题”的概念，这个概念基本相当于库克提出的NP完全性。莱文列出了6个通用搜索问题，其中就包括可满足性问题，从而正式建立了P/NP问题。

在莱文取得的这两项令人惊叹的成就中，仅第二项几乎就可以跟为库克赢得图灵奖的那篇论文媲美。然而柯尔莫哥洛夫却认为两个结果无法分开发表，于是他强迫莱文把它们写在一篇论文里。当时俄罗斯的数学出版物的风格导致精简的论文更受欢迎，文中基本不会给出支持结论的证据的细节。莱文把这种风格发挥到了极致，将两项成果写在了一篇只有两页的论文里。

莱文在苏联申请博士学位的时候，没有把这篇杰作包括在论文集里。当时所有苏联年轻人都是苏联共产主义青年团团员。莱文经常在共青团的活动中调皮捣蛋，他低估了这些行为的后果。这些行为最终导致他没有被授予博士学位，因为政治不合格。

莱文在苏联无法摆脱政治问题的纠缠，他想方设法在1978年移民到了美国。艾伯特·梅耶将他录取为麻省理工学院的研究生，由于他以前的研究工作太厉害，第二年就拿到了博士学位。后来莱文到波士顿大学担任教授，直到今天。

莱文的工作在20世纪70年代中期才被美国所知晓，那时P/NP问题已经广受关注。莱文没能分享库克1982年的图灵奖，而直到20世纪80年代末期，人们才把NP完全性的早期结果称为库克–莱文定理。

20世纪80年代后，苏联与美国的关系开始缓和。一名俄罗斯学生亚历山大·拉兹波洛夫为一种通过电路复杂度理论来证明P≠NP的方法做出了重要贡献，这个故事我们留到第7章再讲。

随着苏联的解体和互联网的崛起，俄罗斯数学研究者的工作不再与世隔绝。计算复杂度领域真正成为全球性的研究课题。

5.3 哥德尔的信

1956年，库尔特·哥德尔写信给约翰·冯·诺依曼，后者是计算机科学和其他许多领域的先驱。在这封（用德语写的）信里，哥德尔讨论了可满足性问题，并用一种不同的术语规范化了P/NP问题。他提出如果我们生活在P=NP的世界里，那么“数学家思考‘是或否’的思想工作将完全由机器来代替……而我个人认为，这完全是有可能发生的”。这封信的写作时间比库克和莱文的论文发表早15年。

我们不知道冯·诺依曼是否回复了这封信，甚至不知道他是否看到了它。冯·诺依曼那时已经身患癌症，不久就于1957年辞世。这封信则直到20世纪80年代才为科学界所知。哥德尔在1978年去世，然而他晚年已经神志不清，无法注意到库克只是重述了他提出的问题。

既然是哥德尔早于库克和莱文发现了P/NP问题，为什么我们不把P=NP称为“哥德尔问题”，以肯定他的贡献呢？因为科学领域遵循的是哥伦布原理：哥伦布出名并不是因为他第一个发现了美洲，而是因为他最后一个发现了美洲。哥德尔本人也不是没有问题。他自己都没有意识到当年向冯·诺依曼提出的这个问题的重要性，不然他不会在之后的任何一个公开场合都没有提出这个问题。库克和莱文仍然是率先发表论文，并向学术界呈现P/NP问题的人。

为纪念哥德尔为逻辑学作出的奠基性贡献以及哥德尔的这封重要信件，理论计算机科学界于1993年设立了哥德尔奖，表彰该领域近期发表的杰出研究论文。

5.4 火星人法则

我们怎么判断一个科学理念是自然产生的（就如同造物主创造的一样），还是由人类的主观活动产生的呢？假设我们发现存在火星文明，其科技水平与我们地球文明的水平相当。如果在火星文明中有一个与地球文明相同或相近的理念，那么这个理念应该是自然产生的，因为它分别来源于两个独立的文明。

当然不存在火星文明来和地球文明做比较，所以只能想象一下。假设火星人的计算机器叫Exigius[1]，它和图灵机有本质的区别，但是两者有着相同的计算能力。那么火星版的丘奇–图灵论题是：一切可计算的事物都能被Exigius计算。所以从火星人的角度看，图灵机不是自然产生的，而计算这个概念是自然产生的。

1 Exigius这个名字来自美国科幻电视连续剧*My Favorite Martian*，中文译为《火星叔叔马丁》。——译者注

对于P/NP问题的自然性，我们不需要火星人来证明。当年苏联和北美的研究者在彼此隔绝的情况下，发现了同一个P/NP问题以及NP完全问题。他们的研究动机不同：在东方，人们试图理解蛮力搜索的必要性；而在西方，人们则希望理解高效能计算的力量。他们沿着各自的研究之路，最终来到了同样的地方，一个在15年前就被库尔特·哥德尔捷足先登的地方。

类似地，假想中的火星人应该也会发现P/NP问题（或者类似的问题，但愿他们还没有先我们一步解决这个问题），并且很可能他们也认为这个问题是自然产生的，而且非常重要。

第6章　处理困难的问题

在第2章，我们看到了P=NP的美好世界。生活将如此轻松，人们可以对一切进行最优化，了解万事万物。我们将拥有让人心想事成的机器。这一切太过美好了，以至于有点吓人，而且几乎肯定是不切实际的幻想。

我们更可能生活在一个相对混乱的世界，一个P≠NP的“不优雅”的世界。即使P=NP，在找到解决NP问题的算法之前，我们还是一样生活在P≠NP的世界。那么对于那些我们不能有效求解的问题，我们该怎么处理？直接放弃吗？

有时候我们不得不处理困难的问题。哈利在Acme制造厂担任主调度师。老板艾米让哈利安排机器的调度方案，来生产Acme最新的手机A-Phone，并要求使用的时间尽可能短。哈利读了本书开头的几章然后回答：“对不起，但那个问题是NP完全的。连很多著名的计算机科学家都不相信这类问题存在有效的解决方法，所以我试也是白试。我还是去打保龄球吧。”艾米直接祝哈利玩得开心，然后不用来上班了。

艾米很快提拔了乔治来代替哈利。幸好乔治读了本书的这一章，然后拟出了一个生产A-Phone的调度方案。乔治写出了一个总能找到最好的调度方案的美妙算法吗？没有。但是乔治有没有把事做完?是的。

NP完全问题是难以驯服的野兽，如果有P≠NP，那我们不可能找到一个能对所有问题总给出最好方案的快速算法。然而我们不必放弃，本章我们就来看看有哪些方法能在处理困难问题时派上用场。

对于中等规模的问题，我们可以用如今非常快的计算机遍历所有的解决方案。我们可以采用一些虽然不能包治百病，但对我们关心的问题起作用的算法。还有的算法虽然不能给出所有可能方案中最好的，但是能给出一个足够好的方案。

有的时候你想尽办法也找不到一个NP完全问题的解决方法。你可以试着解决一个不同的问题，或者干脆放弃。天涯何处无芳草，何必在一棵树上吊死。

6.1 蛮力

计算机很快，甚至是你桌上或口袋里的那些计算机，也快到不可思议，以至于它们可以通过使用“蛮力”搜索所有可能的方案，来解决一个中等规模的问题。

以前可不是这样的。在斯蒂芬·库克将P/NP问题呈献给世人的1971年，英特尔公司发布了它的第一个微处理器芯片——Intel 4004。Intel 4004是第一个包含整个CPU的芯片，也是第一个商业化的微处理器。Intel 4004当时的运行速度是每秒执行92 000条指令。

以库克的可满足性问题为例，设共有20个变量，考虑一个简单的算法，它会遍历并尝试所有让每个变量分别为TRUE或FALSE的解集。如果验证每个解需要100步，那么我们可以在19分钟内判断某个表达式是否能被满足。这个时间不是特别长，毕竟用20个变量表示不了多少内容。

解决一个25个变量的问题需花费10小时，而求解30个变量的问题将超过13天。一个40个变量的问题，如果按1971年的计算能力，将算到2009年。

这段时间里，英特尔每年都会发布很多型号的处理器。让我们从2009年选一个型号，Intel i7-870，它每秒执行的指令数是29.3亿条（是Intel 4004的3万多倍）。用这个速度，求解40个变量的问题将花费大概10个小时。也就是说，如果你在1971年想求解一个40个变量的问题，你可以选择花38年玩自己的手指，然后用2009年的科技解决这个问题，这比从1971年就开始用当时的科技来计算还要快。

针对某些其他的NP问题，算例的规模可以更大。NP完全的旅行推销员问题是指找到访问许多个城市的最短路线。用所谓的分割面法，我们可以用很短时间解决1万个城市规模的旅行推销员问题。美国人口超过500的城市共有13 509个，图6-1是穿过这些城市的最短路径。

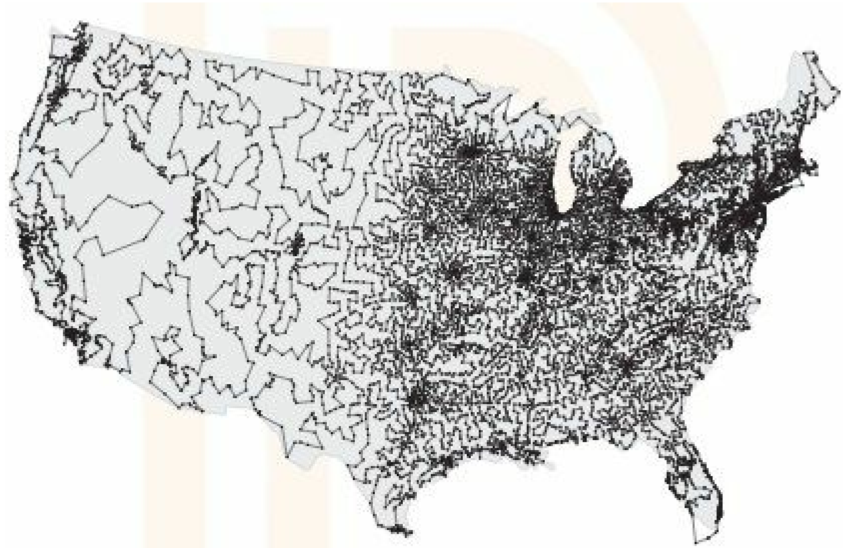


图**6-1　500**人以上城市的旅行推销员问题

对于其他的NP问题，可能性的数目大到连解决中等规模的问题都不太可能。

虽然我们即将逼近计算机速度的物理极限，但随着摩尔定律反映出的芯片上核心处理器个数的快速增长，人们仍然期望计算机的运算能力持续出现戏剧性的增长。这将有助于未来更大规模的NP完全问题得到解决。然而问题的复杂度也会随着规模的扩大迅速增长。所以别指望能在近期解决150个变量的可满足性问题，或是找到2万个城市之间的最佳旅行方案。

6.2 启发式方法

17世纪的伐木工在测量中经常用他们拇指的长度来大致度量1“英寸”的距离。“拇指规则”这一俗语可能就源自于此，意思是在决定某些问题答案的过程中使用一种不太准确但大致可用的简化方法。谚语“朝霞不出门，晚霞行千里”给出了一种粗略但通常相当可靠的预测天气的方式。摩尔定律本身也提供了一种粗略地预测未来计算机的计算能力的方法。

计算机算法同样也是一种过程。启发式方法是一种原理类似“拇指规则”的算法，虽然不能保证给出正确的答案，但可以对大多数你想解决的问题给出答案。人们在认识到某些问题是NP完全问题之前，就给出了它们的启发式解法。几十年来，我们为解决各种困难的问题发明了很多复杂的启发式方法。虽然这些方法不适用于任意一个NP完全问题的所有情况，但有助于根据特定的情况解决人们所面临的部分问题。

让我们来详细分析一个地图填色问题的简单而强大的启发式方法。在第3章我们证明了为美国地图填色至少需要4种颜色，才能保证所有接壤的州具有不同的颜色。

加利福尼亚、俄勒冈、爱达荷、犹他、亚利桑那这几个州围绕着内华达州成为一圈。至少需要三种颜色为加利福尼亚、俄勒冈、爱达荷、犹他、亚利桑那这几个州填色。而内华达州并非个例。肯塔基州周围围绕着田纳西、弗吉尼亚、西弗吉尼亚、俄亥俄、印第安纳、伊利诺伊和密苏里这几个州。同样，你需要至少3种颜色填充这几个围成一圈的州，再用第4种颜色填充肯塔基州。

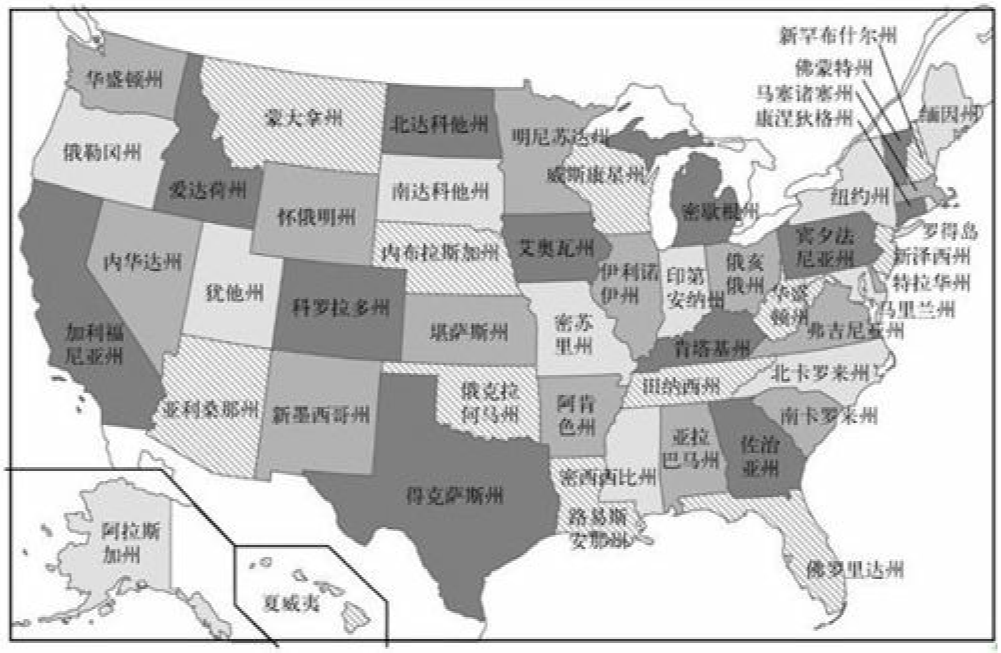


图**6-2**　美国地图

一张地图只要上面的某个地区被奇数个相邻的地区所环绕，那么它最少需要4种颜色。

这里有一张亚美尼亚各州的地图。



图**6-3**　亚美尼亚

只有两个州完全在亚美尼亚内部，不与其他国家接壤。科泰克州（Kotayk'）有6个邻州，而首都埃里温（Yerevan）有4个邻州。每个不靠海的省都有奇数个邻州。所以启发式方法告诉我们，也许能用3种方法为这张地图填色。事实上真的可以。



图**6-4**　填色后的亚美尼亚地图

瑞士有5个邻国：法国、意大利、奥地利、列支敦士登和德国。虽然邻国数为5，我们还是可以用3种颜色填充这张地图。



图**6-5**　瑞士



图**6-6**　填色后的瑞士地图

列支敦士登将瑞士和奥地利的国境线分割为两段，所以对于地图填色这个问题，我们需要把奥地利算两次。重要的不是邻国的数量，而是国境线的数量。瑞士有6段国境线，6是偶数。而且只有外围的国境线才算

数，对于那些完全位于其他国家领土内部的国家（如意大利内部的梵蒂冈），我们不把它们的国境线计算在内。

如果这个启发式方法总能给出正确的答案，那么它将是一个解决NP完全的三色填充地图问题的有效算法，进而我们就有了解决所有NP问题的有效算法，即P=NP。读到这里你可能猜到了，这个启发式方法不总是有效。

令这个启发式方法失效的反例有两个。

1. 地图上有一个湖将两个地区分隔开来，如密歇根湖将伊利诺伊州和密歇根州分隔开来。

2. 地图上有4个以上的地区汇聚于同一点，如亚利桑那、新墨西哥、科罗拉多和犹他这四个州。

在填充美国地图时这些反例不重要，因为我们已经知道填充美国地图至少要4种颜色，这甚至没有把填充湖面的蓝色计算在内。

然而这个启发式方法在现实世界中很少失效。为说明问题，让我们来看看虚构的敌友国地图。虽然每个省都有偶数个邻省，但还是不能只用3种颜色为敌友国地图填色（不包括湖面的蓝色），以确保两个接壤的省的颜色不同。

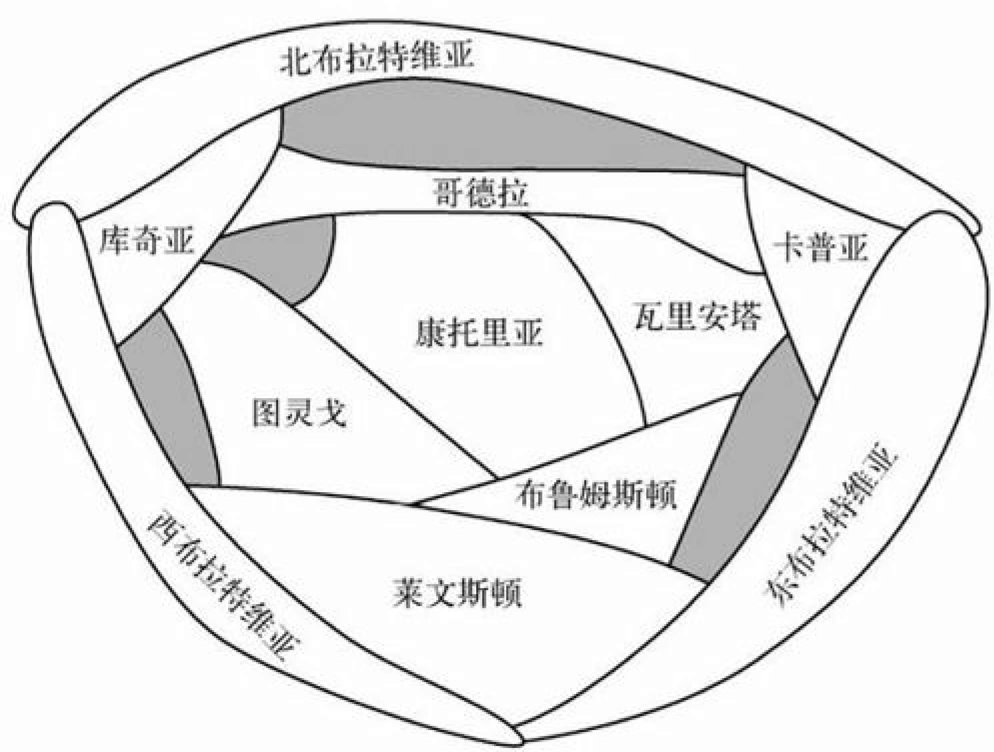


图**6-7**　敌友国地图

每个省都和4个其他的省接壤。没有一个省被邻省围绕成环，因为有湖水从中阻隔。启发式方法告诉我们可以只用3种颜色填充这11个省，但实际上这不可能，只用三色必然导致邻省同色的情况。启发式方法在敌友国能否用三色填充地图的问题中给出了错误的答案。

可满足性理论及其应用国际学术年会（The International Conference on Theory and Applications of Satisfiability Testing）是可满足性问题相关领域研究人士的一个交流和展示的平台，尤其重视优秀的启发式方法。会议组织了SAT[1]竞赛，比拼各种计算机算法求解可满足性问题的能力。竞赛用的可满足性问题有的是随机生成的，有的是特意构造出来以提高难度的，还有的则是实际应用中产生的。这些算法中有许多都能解决某些达到一百万个变量规模的可满足性问题。

1 SAT为可满足性问题（satisfiability）的前三个字母。——译者注

启发式方法并不总能奏效。没人能在SAT竞赛中取得完美的成绩。但通常来讲，对各种计算技巧的巧妙使用，再加上当今最快的机器的强劲火力支援，有些算法甚至能求解规模很大的最优化问题。

6.3　搜索小规模的解

如果我们想在敌友国的2万居民中寻找数目为3的团，最原始的方法需要检查1万亿多一点种可能，这对如今的计算机不太困难。但是数目增长得太快了：4个人的团就存在6000万亿种可能；5个人的团则有26亿亿种可能；6个人的团则有880万亿亿（88后面有21个0）种可能。很快，问题的规模就超出了机器的处理能力。

对于其他NP完全问题，搜索小规模的解可能会简单一些。如果敌友国有一群人满足如下条件：即对任意一对朋友，都至少有一个人在这群人中，那么这样一群人被称为一个舒适组（very cozy group）。

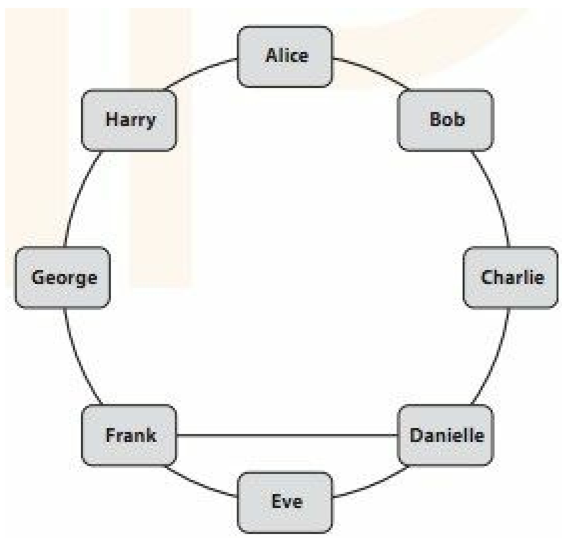


图**6-8**　舒适组

在这个好友关系图中，Bob、Danielle、Frank和Harry组成了一个非常舒适的组，因为每对好友关系中至少有Bob、Danielle、Frank和Harry中的一个人。

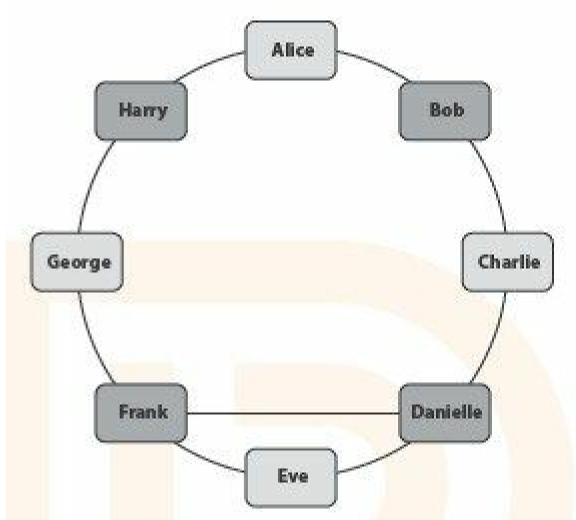


图**6-9**　包含**4**个人的舒适组

在这个好友关系图中，不存在3个人的舒适组。例如，如果我们把Alice、Charlie和Frank这3个人看做舒适组，那么就漏掉了Eve和Danielle，以及George和Harry这两对朋友。

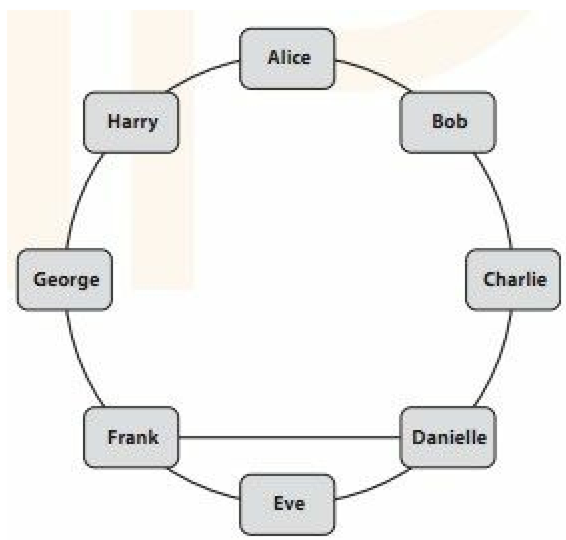


图**6-10**　包含**3**个人的舒适组

舒适组问题，即人们熟知的“顶点覆盖问题”，是卡普原来列出的NP完全问题之一。

我们来看图中的Frank。如果Frank不在舒适组中，那么George、Danielle

和Eve必须在舒适组中，因为他们都和Frank是朋友关系。如果Frank有100个朋友，则要么他自己在舒适组中，要么他所有的朋友都在舒适组中。

使用这样的技巧，计算机科学家可以不用穷举就能找到小的舒适组。找到5个人的舒适组只需要检查大概100 000种可能。找10个人的组则需检查200 000种可能。30个人的组则是601 000种——所有这些用你的笔记本电脑来算基本上瞬间就能完成。搜索包含113人的舒适组需要检查1万亿种可能，其计算时间基本等于搜索3个人的团的时间。

等一下！卡普不是证明找到最小的舒适组也是NP完全问题吗？看起来寻找舒适组并不怎么困难嘛。想想敌友国有多少对好友关系。这么多的好友关系，舒适组根本不可能只用区区113人就保证每对好友中至少有一个人在其中。当舒适组的人数达到合理的规模以后，相应的必须检验的可能性的规模会变得非常大。

如果你寻找的是150人的组，有1500万亿种可能需要搜索；200个人则是10万亿亿种（10^{21}）；而500人，则大概是38后面跟51个0。找到敌友国中最小的舒适组基本上是不可能完成的任务。但如果我们只是在100个人中寻找有没有舒适组，这个任务还是可以在合理的时间内完成的。

6.4　近似计算方法

也许我们没有问题的最佳答案，但通常一个不坏的答案就足够好了。

比如NP完全的旅行推销员问题，即找到访问多个城市的最短路线。

如果我想到50个城市旅行，最短的路线是180万英里，而我已经找到了一个需要181万英里的路线，通常大可不必为多走的1万英里劳心费神。

换一个角度讲，如果走1英里要花费1美元，而我旅行的收入是180.5万美元，那么180万英里和181万英里的差别，可能就是挣5000美元和赔5000美元的差别。如果我对旅行计划作出一点点的改进，比如180.3万英里，那我就可以扭亏为盈了。

虽然地图上的旅行推销员问题是一个被认为难以有效解决的NP完全问题，但是我们可以找到非常接近最优解的旅行路线。桑吉弗·阿罗拉和乔·米切尔给出了一个算法，原理是先将地图切分为许多小块，然后在所有小块中找到小段的最佳路线，最后通过一种高明的方式将这些小段的路线连接起来。

如下图中的71 009个中国城市。



图**6-11**　中国城市

我们把一个细密的网格套在地图上，然后在每个小格中求解旅行推销员问题。如果某个小区域的城市太多，那就进一步把该区域细分成更小的网格。

用这种技术，我们可以在合理的时间内，找到与最优方案只差百分之几的方案。

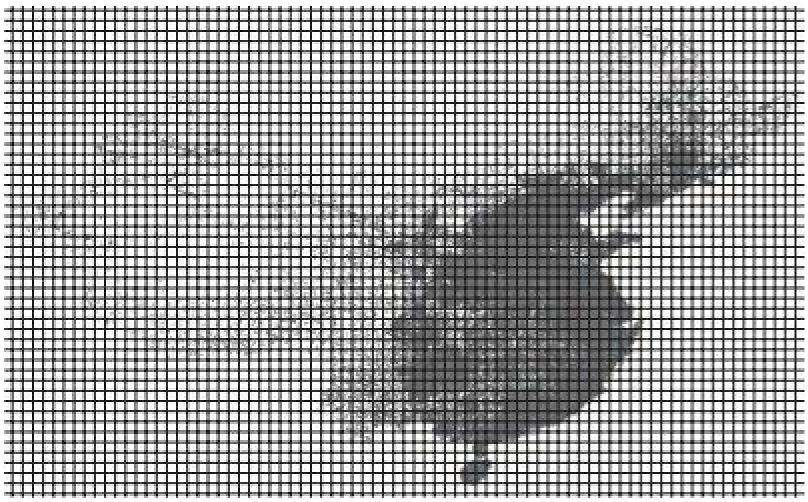


图**6-12**　中国城市的网格

如果所有NP问题都有这么好的近似计算方法，那么P/NP问题几乎就不重要了。然而生活没有那么轻松。比如团问题，即找到一大群互为朋友的人。目前还不存在任何通用的方法可以估算最大团的问题。敌友国可能存在2000人的团，但在合理的时间内，我们连15人的团都找不到。

如果P=NP，我们就能有效地找到全局最大团。这被证明是能稳定地找到团的唯一方式，哪怕是寻找人数是最大团的0.1%的团，也只能采取这种方式。发现任何能比这更好地通过估算来解决团问题的算法，都将令P=NP，即有效地解决团问题本身。

许多NP问题既不像团问题这样难以估算，又不像地图上的旅行推销员问题那样容易估算。让我们回过头再来看看舒适组问题。

像这样一个小规模的例子，我们通过穷举就能得出最小的舒适组有4个人，例如Frank、Danielle、Harry和Bob。假设我们不知道这一点，让我来描述一个估算最小舒适组的简单的算法。

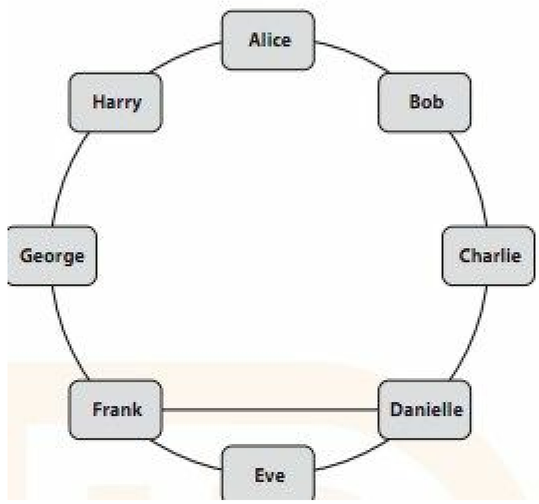


图**6-13**　舒适组**II**

首先任选一对朋友，比如Alice和Harry，并标记他们。

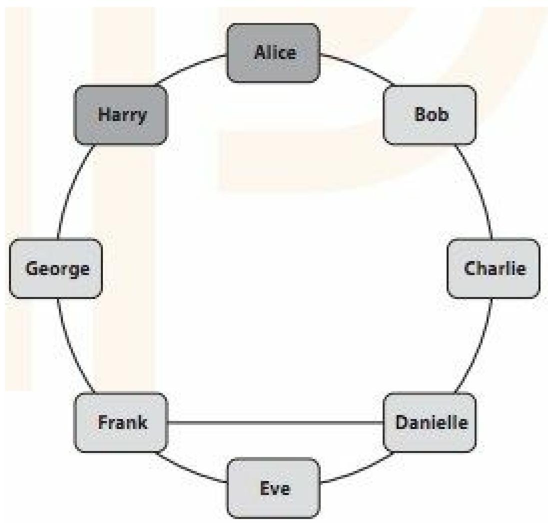


图**6-14**　舒适组**II**，**2**人被标记

找到另一对没被标记的朋友，然后也标记他们。

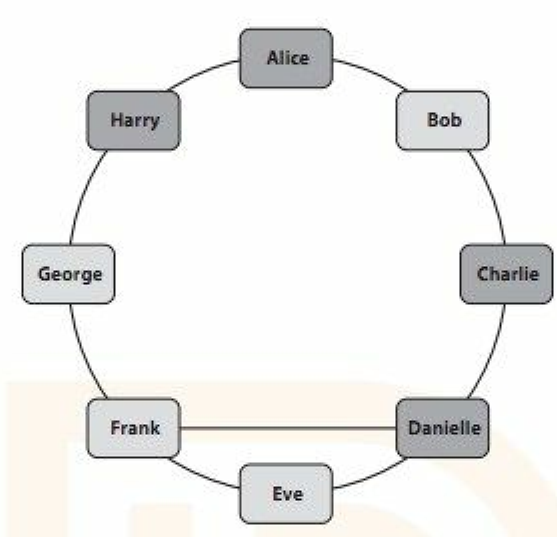


图**6-15**　舒适组**II**，**4**人被标记

重复上一过程直到找不到未被标记的朋友。

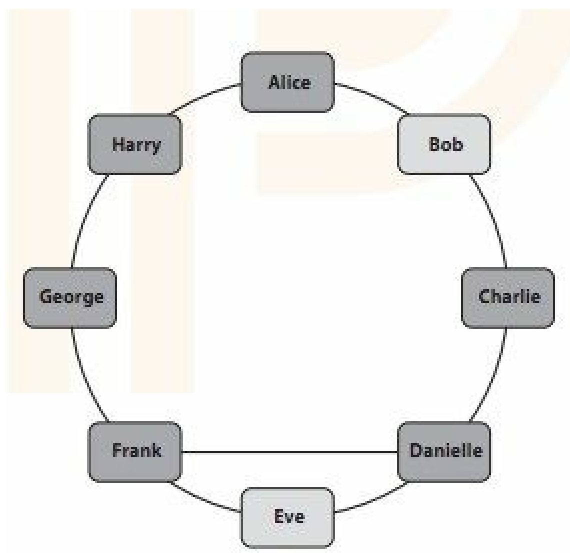


图**6-16**　舒适组**II**，**6**人被标记

最后，所有被标记的人组成的将是一个舒适组。

在这个示例中，我们得到了一个6人的舒适组。每个舒适组都要从被标记的3组好友中每组至少选一个人，所以人数不能少于3人。那么最好的舒适组的人数应该在3到6之间。

这个算法适用于所有的好友关系图。如果我们的算法找到了人数为100的舒适组（即总共添加了50对好友），那么我们知道最好的舒适组应该在50到100人之间。

我们总能找到一个舒适组，它的人数是最小舒适组人数的2倍。我们能得到比这好很多的结果吗？也许不能。

如果P=NP，我们就能轻松找出最小的舒适组。但如果P≠NP呢？一般地讲，不可能找到比最小人数多36%或更小的舒适组。任何能找到比最小人数多36%的舒适组的算法，都能被转化为一个可解决所有NP问题的算法。这个转化的过程与一系列在1990年到2005年之间证明的艰深结论有关。

上面介绍的简单算法通过逐个添加一对好友，能找到最多两倍于最小人数（也就是比最小人数多100%）的舒适组。如果P≠NP，那么我们能希望的最好结果就是找到比最小人数多36%的舒适组。我们有可能找到比最小人数多50%的舒适组吗？

为解答上述问题和其他相关问题，计算机科学家萨布哈什·霍特引入了一种他称为独特游戏（unique games）的问题，该问题是地图填色问题的一个变种，增加了一些约束相邻地区填色方式的规则。独特游戏猜想是说独特游戏是NP完全的，人们至今无法判断该猜想是否为真[1]。

1 关于独特游戏和独特游戏猜想，参见维基百科页面
http://en.wikipedia.org/wiki/Unique_games_conjecture。——译者注

如果独特游戏猜想是真的，我们甚至可以进一步约束近似计算的能力。霍特证明，如果该猜想为真，那么不能找到比最小人数的2倍只少一点的舒适组。如果该猜想为真，我们不能得到比上述简单算法好很多的结果。

6.5　解决一个不同的问题

有时候不论多么高明的技巧，都对某个NP问题无能为力。那么，我们可以试着解决另一个不同的问题。

简是帕罗奥图市一家名为图灵之家（Chez Turing）的餐馆的主厨，她想研发一种新的番茄酱来搭配她的名菜——砂锅意面。图灵之家是一家引领风潮的新派饭店，它开创了计算派菜系。简不直接使用食物做试验，而是向计算机输入她想要的颜色、味道、气味和口感，然后计算机就会搜索各种食材的组合，精确调配出所需的口味，这样研发的新菜不仅好吃，并且降低了制作成本和所含的热量。这一次，简想要研发一种深红色的酱，辛辣等级为5，口感类似燕麦但更顺滑，能轻触味蕾5和11，并能对砂锅意面本身的味道产生其他各种微妙的影响，以达到完美的进食体验。

计算机没能找到符合她所有要求的食材组合方式。于是简打电话给计算机天才汤姆，他答应过来帮助简，顺带赚取一顿免费大餐。汤姆尝试了他知道的所有启发式方法，也用了他查到的几种新方法。他还到云端，租借了Amazon机器上的计算时间，试图用真正生猛的计算能力来攻破问题。而那也没有奏效。他接着去拜访湾区的所有朋友，并宣布为第一个找到制作酱料方法的人提供一个令人垂涎的图灵之家的预定餐位。一周后，所有人都表示放弃，因为这个问题实在太难解决了。

汤姆回去告诉简这个坏消息。这是他第一次没能找到简需要的食材和制作方式。汤姆问："能不能试试用其他方式来做酱料，比如稍微改变一下味道？"简不肯让步，认为酱料必须要具备这种味道和口感，否则会破坏砂锅意面在口中的感觉。汤姆又问："那砂锅意面本身呢？"

既然为这种特定的砂锅意面找到合适的酱汁是一个很难的计算问题，简和汤姆转而搜索新的能够完美契合的砂锅意面和酱汁。这个任务稍微容易计算一些，计算机在数小时内就输出了一份食物配方。通过改变要解决的问题，图灵之家又有了一道新的招牌菜。

在另一种情境中，改变要解决的问题则令计算机安全专家们头痛不已，他们开发出的最新加密技术能让交易变得更加安全，前提是坏蛋只用专

家们预想到的方法来破译密码。但是坏蛋们往往不按套路出牌，却偶尔能歪打正着。

智能卡表面上和信用卡一样，但里面内嵌了一个小小的处理器，存储着一个密钥。智能卡可以用来识别身份，也能存少量的钱，这样你在商店购物或者泊车咪表处交费时，就不用和集约化的计算机进行连接和通信也能完成交易。即使有人能截取到智能卡的输出和输入数据流，也很难猜出密钥。

假如托马斯在酒店工作，趁安妮去游泳的时候拿了她的智能卡。托马斯有一个特殊的读卡器，用它能试着给智能卡发送一些数据，然后看看返回了些什么数据。如果P=NP，那么窃贼能根据这些输入/输出行为找到密钥。而我们更有可能生活在P≠NP的世界里，找到密钥是很困难的。智能卡数据交换协议是经过精心设计的，本章提到的蛮力搜索或启发式算法等技术对于找到密钥都收效甚微。如果找不到密钥，托马斯通常只能把卡还给安妮，或者当安妮发现卡片丢失，直接给银行打电话注销这张卡。

而托马斯不是一般的身份窃贼。他能以设计者意想不到的方式来“使用”智能卡。他把卡扔到微波炉里加热几秒。你可别在家尝试这个，微波炉的能量会让智能卡或任何其他电子设备接近报废。即使是很短时间的微波加热处理，也可能导致某些电路的短路，即使不是永久性地损伤智能卡，也会让它的输入/输出行为变得无法预料。

无法预料的行为正是托马斯想要的。如果智能卡正常工作，他也许不知道如何找到密钥。而一个略微受损的智能卡虽然仍可以对输入的数据做出响应，但是输入/输出的行为也和原来不一样了。

新的输入/输出行为仍与密钥有关，但是原来那种令密钥难以被找到的设计失效了。这时如果托马斯再使用启发式或蛮力搜索等方法来寻找密钥，就有可能奏效。

一旦托马斯找到密钥，他就能用它制作两个和安妮原先正常的卡片一模一样的智能卡。托马斯把其中之一还给安妮，另一个留给自己。托马斯如果足够小心的话，就可以从安妮的账户偷钱，而安妮和银行可能要等几周或几个月后才会发现问题。

托马斯找到密钥的方式是，把一个被故意设计成难以解决的NP问题转变为一个完全没被设计过的问题。新一代的智能卡已经拥有了对抗微波攻击的机制。但我们绝不能低估那些一心想找到秘密信息的家伙们的想象力。

6.6 接受现实

有时候一个问题天生排斥任何可能解决它的方法，对此你能做的只有放弃，然后去干点别的。

理查德偶然发现了一种超级药物的化学式，这种药能让他主宰世界，或者至少是主宰大辛辛那提区。他制定了一个向米勒市水处理厂投放药物的计划，喝了水的居民将失去警惕，变得非常容易被操控。到那时，理查德就能“挺身而出”，接管他们的社会。

只要收集到足够多的合适的化学品，理查德就能实现他的梦想。不幸的是，以前发生的某次事件导致政府禁止了供应商直接向任何个人出售这些化学品。于是理查德开始分析如何分解各种日用品，然后从原料中制备这些化学品。理查德往笔记本电脑里输入了各种参数：购买日用品的成本，从大辛辛那提区的20个沃尔玛和14个Targets超市能够安全购买而不引起怀疑的每种日用品的数量，运输和仓储的成本（他需要好几个仓库来放买来的东西）。有好多种日用品的组合可供选择。他把这些参数输入电脑，但是需要检查的可能性数量太多了，计算机无法找到符合他的时间和资金限制的方案。他打心眼里觉得一定存在满足所有条件的方案，但不管他怎么尝试，用尽了所有他从网上能找到的启发式方法和估计算法，就是找不到这样一个能凑齐所需化学品的方案。最终理查德放弃了，回到他在本地牙膏厂的保安岗位上。

就这样，NP问题的复杂性拯救了辛辛那提。

6.7 总结

通常，只一种技术不足以处理人们面临的困难的NP问题。我们经常需要尝试本章提到的几种技术。如果无法解决需要解决的问题，那么可以尝试解决一个不同的问题。如果新的问题仍然是NP完全问题，那我们还可以用某些启发式方法来对它发起进攻。启发式方法很少能完全地解决问题，但是它们可以给出足够好的近似解决方案。

如果P=NP，所有这些问题都将烟消云散，我们将用一个简单的算法做很多不简单的事情。但即使事实如我们料想的那样（即P≠NP），我们还是几乎总能收获点什么。可能过程中需要付出更多的努力，可能最终解决的问题和最想解决的问题不太一样，可能根本无法得到最好的解决方案。但是，如果我们仍然能把事情做完，那就足够好了。

第7章　证明P≠NP

尤里斯·哈特马尼斯是计算复杂度理论的奠基人之一，他曾说：“我们都知道P和NP不一样，但就是不知道如何证明这一点。”

我们在前面几章中从多方面探讨了P/NP问题，包括它是什么，它的研究意义，P=NP的虚幻美妙世界，以及如何处理由P≠NP带来的困难问题。

P/NP问题也是一个迷人而富有挑战性的数学问题。从库克、卡普和莱文将这个问题及其重要意义呈现给世界的那一刻起，计算机科学家和数学家就一直努力形式化地证明P=NP或P≠NP。但所有常规的手段都失效了。到20世纪70年代末，大家达成的共识是“P/NP问题的解决可能有赖于大幅度创新的证明技术”。



图**7-1　DILBERT © 1997 Scott Adams**。使用需得到**UNIVERSALUCLICK**的许可，版权所有，侵权必究

接下来的几十年中，计算机科学和数学都取得了不可思议的进步，包括解决了所有未解决数学问题中最著名的一个——费马大定理。1637年左右，法国律师兼数学爱好者皮埃尔·德·费马在他的《算术》（一本古希腊教科书）的书页边缘上写下如下这样一段文字：

> 将一个立方数分成两个立方数之和，或一个四次幂分成两个四次幂之和，或更一般地，将一个高于二次的幂分成两个同次幂之和，都是不可能的。对此我确信已发现了一种美妙的证明方法，可惜这里空白的地方太小，写不下。

也就是说，不存在大于0的三个自然数a、b和c，以及大于2的自然数n，

能令$a^n+b^n=c^n$。费马再也没有提到这个定理的证明方法，也有可能他压根就没想出正确的证法。后来这个问题闻名于世，成为一个经典的数学未解之谜。我小时候也梦想成为第一个解决这个著名问题的人。另外一个和我有着共同梦想的小孩长大后实现了他的梦想。1994年，普林斯顿大学的数学家安德鲁·怀尔斯在前人发表的一系列有关数论的论文基础上，构造出了费马定理的证明。他立刻成了名人，至少拥有了作为一个数学家所能够享有的无上荣耀。

这一章不会告诉你如何解决P/NP问题，否则这将是一本非常不同的书。我们只会介绍一些人们在尝试解决P/NP问题时曾经产生过的想法。可惜，这些想法没能让我们距离解决该问题更近一点。要证明P≠NP，就要证明不存在能解决某些NP问题的算法（甚至包括那些未被发现的算法）。很难去证明不可能做成某件事，尽管这在逻辑上并非不可能。所以，对于这个也许是所有数学问题中最为重要同时最具挑战性的问题，我们还是希望看到它被解决的那一天。

7.1　骗子悖论

请考虑如下这个令人困惑的命题。

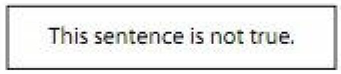


图**7-2**　方框中的命题：这个命题不是真的

这个命题是真是假？如果它是假的，那么它不能不是真的，双重否定意味着它是真的。如果这个命题是真的，那么如其所说，它不是真的而是假的。两种假设，都会导致矛盾。这个悖论经常被称为骗子悖论。我现在正在撒谎。你觉得呢？

良好的数学系统中是不允许出现正确的悖论的，悖论的存在说明数学系统不够好。“这个命题不是真的”是不可被数学规范化的，因为一个命题不能讨论它本身的真伪。

20世纪30年代，库尔特·哥德尔发现了数学可以讨论证明的真伪，他可以构造一些数学命题，来表明其他的命题是否存在能证明其正确性的证据。哥德尔发现了如何让一个命题讨论它本身正确性的证明是否存在的方法，并建立了一个与上面类似的数学公式：

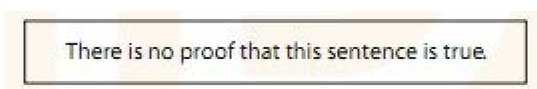


图**7-3**　方框中的证据：没有证据表明这个命题是真的

假设这个命题为假，那么有证据表明这个命题是真的，这和它为假的假设矛盾，所以这个命题必须是真的。

我们又遇到悖论了吗？不完全是。这个命题是真的，然而没有证据表明它是真的。哥德尔看似轻描淡写的一击，却撼动了数学大厦的基础：存在人们不能证明其为真的真命题。[1]

1 我们刚才没有证明那个命题是真的吗？不完全如此，因为不得不首先假设所有能被证明为真的命题都确实是真的。于是哥德尔同时也表明了，我们不能证明“所有能被证明为真的命题都确

实是真的”，除非我们能证明假的命题。这就是你读脚注的好处。

如果有人告诉你他有一个能预测未来的魔盒。你就问他你会用右手打他还是用左手。如果盒子说左手，你就用右手打那个人。如果盒子说右手，你就用左手。不管怎样盒子都是错的。

计算也同样可以这样。我们都见过计算机屏幕上出现一个代表忙碌的小沙漏，不知道这是代表计算机死机了，还是在进行长时间的计算。用户该马上重启机器呢，还是再等一会儿？如果能有一个算法，告诉我们计算机是不是陷进了某种无穷循环该多好！那是很好，但那也是不可能的。

为理解其中的道理，我们先来看一下1936年阿兰·图灵在开创计算机科学的论文中给出的一个例子。计算机程序也是数据，就像文档和报表一样。程序可以分析程序的代码。

一个计算机程序要么最终结束计算并输出结果，要么永远算下去。假设我们有一个算法，它能告诉我们某个程序是否会最终结束。我们把这个程序对它本身运行一下。

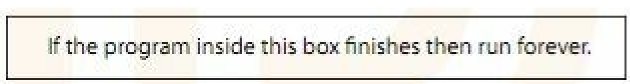


图**7-4**　方框中的程序：如果这个方框中的程序结束，那么它会永远运行下去

这个方框中的程序要么会结束，要么不会结束。不管怎样，我们都遇到了矛盾。所以我们的假设是错误的，即不存在一个程序，能告诉我们某个程序能否最终结束。PC上没有，Mac上也没有。现在没有，100年后也不会有。不存在一个程序能告诉我们某个程序能否最终结束，就这么简单。

我们能用类似的思路来证明人们无法高效地解决某些问题，也就是证明该问题不属于类型P（即我们能高效解决的问题）吗？这确实是可行的。

我们先构造一个算法Q，将一段程序R的代码作为其输入数据，它的工作原理如下：

- 如果程序R在使用R的代码作为数据时能很快输出“Yes”，那么Q输出“No”；
- 否则Q输出“Yes”。

假设算法S是一个高效的算法。那么，Q(S)当且仅当S(S)输出“No”的时候输出“Yes”。所以不存在行为与Q完全相同的高效算法。

Q仍然是一个符合规范的算法。所以Q所解决的这个问题是可被计算的，但不能被高效地计算。

如果我们能证明Q在NP中，也就是说，可以高效地验证其解的有效性，那么Q在NP中，而不在P中。这意味着P≠NP，从而解决了那个伟大的问题。

但是我们不知道，而且通常不认为，存在一个关于Q的NP算法。由于这个原因以及其他的某些原因，这条试图以悖论来解决P/NP问题的道路注定会通向失败，至少它不能直接用来证明P≠NP。

7.2　电路

现代计算设备的核心都有一块集成电路板。

集成电路由上百万或上百亿个微小的晶体管组成。晶体管是一种能放大并控制电子脉冲的元件，它们实现了逻辑门（gate），即一些构建在带有电荷的导线上的简单逻辑操作。

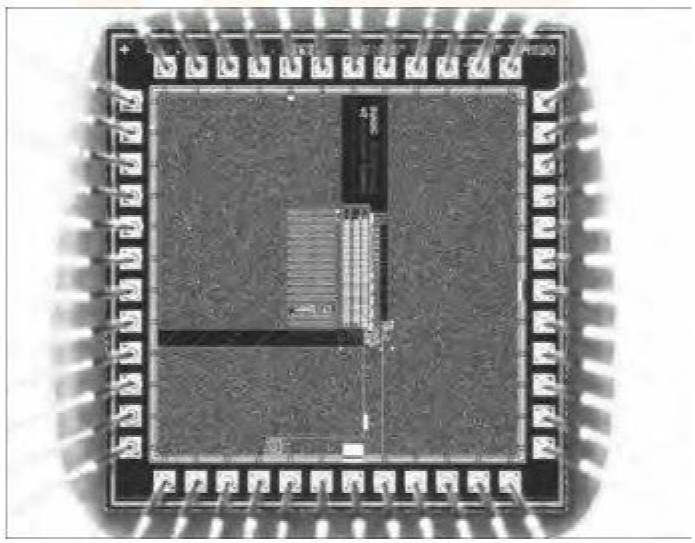


图**7-5**　电路，图片由宾夕法尼亚大学电子和系统工程系提供

我们首先来看简单的导线。每根线要么带有高电压，要么带低电压，只能取两个值中的一个，这经常被表示为开和关、1和0，或是真和假。我们将这些二元系统称之为比特（bit），即“二元数字”（binary digit）的简称。

一根导线做不了什么，几根在一起也能力有限。为了构成计算能力，我们要对导线上的信息进行一些逻辑操作。其中最简单的就是翻转导线的值，这叫做非门（NOT gate）。



图**7-6**　非门

如果非门的左边有电压，那么右边就没有，反之亦然。

计算的真正力量并不是对单个导线的操作，而来自于将多条导线的值以某种机制合并。与门（AND gate）将2条或更多条导线的值合并为一个值，该值仅在所有导线的值都为真时为真。或门（OR gate）将2条或更多条导线的值合并为一个值，该值在至少有一条导线的值为真时为真。

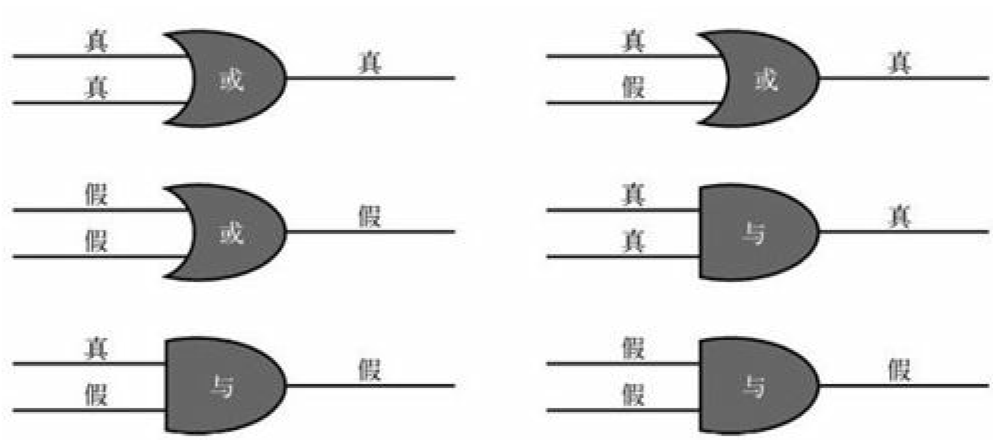


图**7-7**　与门和或门

我们能用这些简单模块构建更复杂的逻辑操作。比如，两条导线做异或操作（Exclusive-OR）的值只在仅有一条导线的值为真时为真。

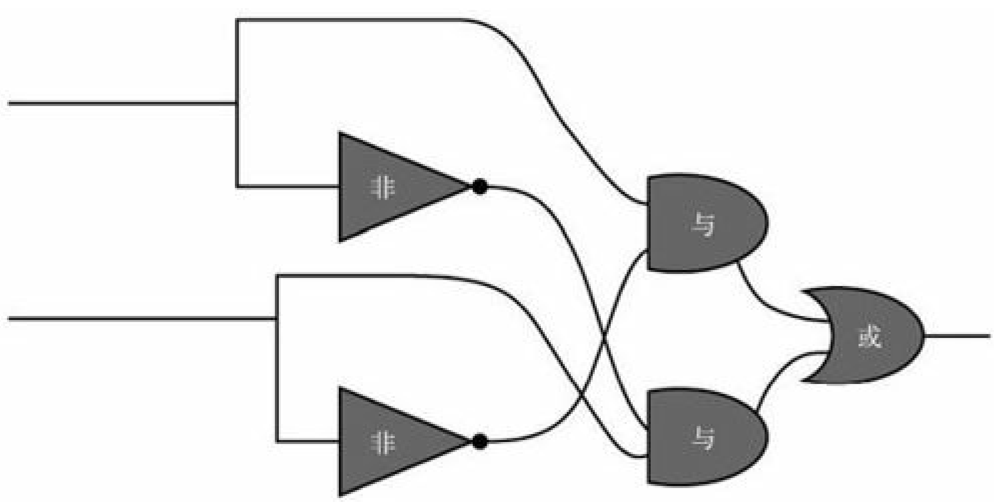


图**7-8**　异或门

每一个函数，无论它多复杂，都能用与、或、非三种门组成的电路来计算。让我们回顾一下敌友国的团问题，即我们想找到一群彼此互为好友的人。我们可以用一个与门来表示Alice、Bob和Carol是否互为朋友。

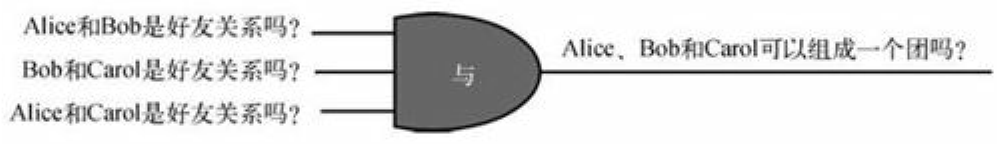


图**7-9**　与门

假设我们想知道在Alice、Bob、Carol、David和Eli中是否存在3个人的团。我们用图7-10中的电路来计算这个问题。

这个电路有10个与门，表示的是从5人里面找出可能成团的3个人，共有10种可能。假设我们现在要从敌友国的2万人中找出50人的团。直接转换的电路将有

3 481 788 616 673 927 174 126 704 329 430 033 822 428 741 136 969 800 292 509 234 146 086 195 855 824 730 457 281 170 250 134 309 383 506 694 008 809 661 825 431 661 561 845 957 650 386 210 936 569 600

个与门，规模大到无法实现。

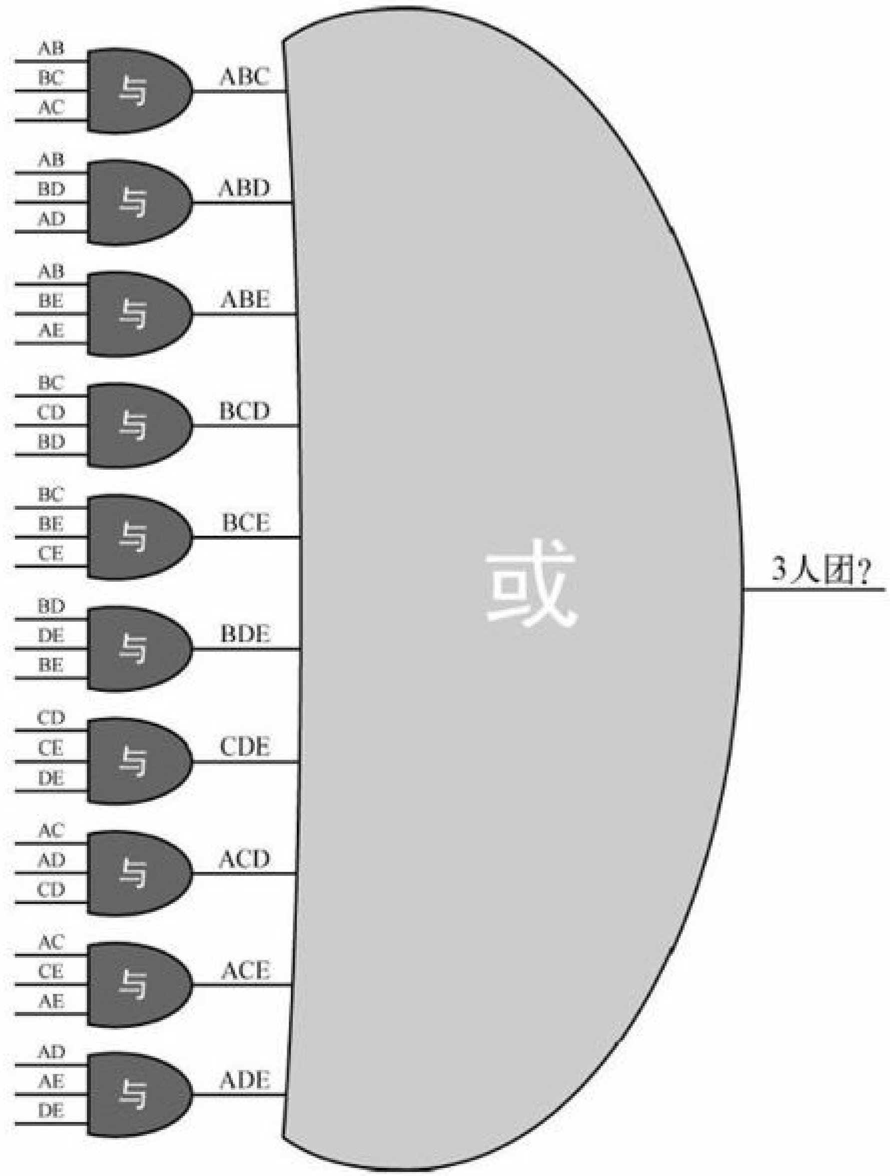


图**7-10**　团问题的电路

这和P/NP问题有什么关系呢？每一个P中的问题（即每一个能被高效地求解的问题），都对应一个规模相对较小的由与、或、非门构成的电路，并且该电路能够求解该问题。如果我们能证明某些NP中的问题（例如团问题），不能对应一个小规模的电路，那么就证明了NP≠P。

那是否能证明团问题不能对应小规模的电路？这个问题和P/NP问题密切相关，同样也是未知的，尽管大部分计算机科学家都相信团问题不对应小规模的电路，正如他们相信P≠NP一样。

看看我们上面为团问题构造的电路。注意其中没有非门。不是所有的问题都能对应不含非门的电路，比如简单的异或操作的电路就必须包含非门。然而，团问题就可以用只有与门和或门的电路来计算，尽管这些电路的规模极为庞大。

1985年，莫斯科斯捷克洛夫数学研究所的一名学生亚历山大·拉兹波洛夫证明，如果电路只包含与门和或门（没有非门），那么任何能求解团问题的电路必须使用数量极多的逻辑门。这并没有解决P/NP问题。因为虽然可以在求解团问题的电路中不使用非门，但说不定使用非门将显著降低电路所需的逻辑门的数量。

尽管如此，拉兹波洛夫的工作曾被看成是有望解决P/NP问题的突破性进展。我的博士生导师迈克尔·塞普斯当年认为P/NP问题的解决指日可待。需要做的只是对拉兹波洛夫的方法稍加改进，让它也适用于非门。如果成功，那么作为NP问题之一的团问题，其求解电路必须要使用数量极大的与、或、非三种逻辑门。所以团问题没有高效能的算法，因为所有高效能的算法都可转化为小规模的电路。这样一来，团问题属于NP但不属于P，从而证明P≠NP。不幸的是，生活并没那么简单。

拉兹波洛夫发表的那些论文是用俄文写的。在论文到达美国后，塞普斯召集了几个苏联学生，大家仔细地进行翻译，并且盼望着拉兹波洛夫的下一篇论文能给P/NP问题画上一个圆满的句号。拉兹波洛夫后来发表了更多优秀的论文，但其中没有一篇是给出P≠NP的证明的。

在第3章我们介绍过敌友国的配对问题，即按好友关系为敌友国的一些居民牵线搭桥。配对问题和团问题一样，也对应只使用与门和或门的电路。证明团问题对应需要大量与门和或门的电路所用的方法，同样也适用于配对问题。任何只用与门和或门解决配对问题的电路，都需要极大数量的逻辑门。

和团问题不同，我们有解决配对问题的高效算法。所以，配对问题存在使用与、或、非三种逻辑门的小规模电路。非门对于求解配对问题不是必要的，但是使用非门会显著降低电路所使用的逻辑门的数量。看上去十分低级和简单的非门，竟然蕴含着巨大的力量。

这个事实让那些企图利用拉兹波洛夫求解团问题的方法来解决P/NP问题的尝试遭受了极大的挫折。即使证明了某个问题在只使用与门和或门时

电路规模很大，也并不能保证在引入非门后电路的规模仍然会很大。后来拉兹波洛夫的工作澄清了这方面的问题。他明确地指出了他的证明在考虑到非门时变得分崩离析，并且无法修补。

后来，拉兹波洛夫和卡内基梅隆大学的史蒂文·鲁迪赫引入了“自然证明”（natural proof）的概念。自然证明涵盖了电路的各种证明方法中的很大一部分，并给出了很强的证据表明不可能用这些方法解决P/NP问题。

同时，使用非“自然”的、基于本章前面提到的悖论思想的方法来构造电路的尝试，也取得了一些小小的进展。然而试图通过“证明某个NP问题需要大规模的求解电路”的方法来找到P/NP问题的答案，可能性已经变得微乎其微。

7.3 证明**P≠NP**时常犯的错误

2010年8月6日，惠普实验室的科学家维纳里·德奥拉利卡向22位顶尖的理论计算机科学家发送了他写的论文，题目简洁有力："P≠NP"。曾经有许多追逐名利（克雷数学研究所提供的百万奖金）的人给出了P/NP问题的各种"证明"，他们认为P≠NP或P=NP，或者不可能判定P是否等于NP，或者声称P/NP问题根本就毫无道理。每年都会涌现很多这类证明的手稿，有的被发送到计算机科学家的电子邮箱里，有的被提交到学术期刊，还有的直接公布在互联网上。最著名的计算机科学期刊持续地收到声称解决了P/NP问题的来稿，后来，该期刊宣布了一条对这些文章的特殊规定。

> *The Journal of the ACM*（JACM）经常收到来稿，声称解决了复杂度理论领域一个长期悬而未决的问题，即P/NP问题。这些来稿严重占用了JACM自愿提供的编辑和审查资源，因为需要一定的时间才能辨识其中的错误。JACM仍然对P/NP及相关问题的解决保持开放的心态，并将继续欢迎针对这个主题的投稿。然而，为了减轻由屡次重复提交并逐一订正错误而带来的负担，本刊决定采取如下规定：对于P/NP这个复杂度理论领域长期未解决的问题或与之相关的主题，任何作者在24个月的周期内只能提交一篇文章，除非受到主编的约稿。此规定也适用于之前被拒稿件的再次提交。

在这些对解决P/NP问题的尝试中，大部分要么不知所云，要么明显是错误的，学术界对此基本上视而不见。而德奥拉利卡的文章则受到了特殊的待遇。他本人在过去曾发表过研究论文，并且这篇论文的写作水平超出了大部分的P/NP投稿。于是有的理论学家感到有必要仔细研读一下这篇论文。这在互联网上引起了轩然大波，一些微博和博客甚至急不可耐地提前宣布P/NP问题得到了解决。几个著名的计算机科学家和数学家在仔细研读了德奥拉利卡的论文后，发现了其中的几处缺陷，有一些是小疏漏，另一些则是大错特错。8月16日，也就是论文公布后的第10天，纽约时报发表了一篇文章来描述事件的整个过程，标题是"第1步：发布模棱两可的证明；第2步：欣赏烟火表演"（Step 1: Post Elusive Proof. Step 2: Watch Fireworks）。此时学术界的一致意见是该证明是错误的，且没有修改的余地。P/NP问题的状态仍然是悬而未决。

希望这本书能让许多读者了解到P/NP问题的重要性，并且对解决这个问题跃跃欲试。我鼓励你试试，因为如果不亲自尝试，很难真正理解一个问题是多么困难。要注意本书并未正式地定义P/NP问题，如果读者真的想求解这个问题，首先要了解其最准确的表述。本书的网站上提供了几个很好的资源，供读者了解P/NP问题和各种求解尝试的技术细节以及公理化描述。

假设你真的找到了P/NP问题的解决方法，该怎么通知克雷数学研究所，让他们把一百万美元的支票寄过来呢？先别急。你几乎不可能有合理的证明。通过了解为什么你的证明可能不行，你说不定会得到启迪。

下面我就描述一下那些认为自己证明了P/NP问题的人通常会犯哪些错误。

早在1550年，意大利数学家吉罗拉莫·卡尔达诺给出了可能是第一个P≠NP证明的反面例子。卡尔达诺被认为是概率论领域的奠基人之一，在宣传他发明的一种新的加密系统时，他声称该系统绝对安全，因为密钥太多，无法逐个检查。实际上这个系统非常容易破解，因为对加密的消息做解密分析时根本不需要检查所有的密钥。

类似卡尔达诺所犯下的根本性错误，在现在证明P≠NP的尝试中屡见不鲜。再次考虑团问题。一类证明的思路是，任何试图求解团问题的算法在找不到一个有效解时，都必须确保不可能存在给定人数的团。唯一可行的方式是检查所有可能的人组成的集合，然后检查给定的集合是否成团。由于集合太多了，任何算法都必须花费特别长的时间。所以团问题不存在高效的解法。故P≠NP。Q.E.D.（Q.E.D.是拉丁短语quod erat demonstrandum的缩写，意为“要展示的就是这么多”，常用于数学证明的结尾，表示证明完毕。）

这种思路有很多的变种。而其中的逻辑漏洞在于“唯一可行的方式”。算法可行的方式也许非常难以捉摸。它并不需要按你认为的方式工作，甚至完全不用遵守问题本身的内在结构。也许有一个算法，把好友关系图转化成某些奇怪的代数表达式，当且仅当团存在时才存在某一类型的解。虽然这不太可能，但是不能因为你认为算法不这样工作，就排除它存在这样的可能性。

很难让使用这种证明技巧的人相信他（或者她，但是给出这些糟糕证明

的人几乎无一例外全部都是男性）是不对的。他们总会反过来要我写出一个不用穷举就能解出团问题的算法。我当然写不出来，能写出来不就意味着P=NP了嘛，况且这不太可能是真的。但证明的负担在你身上，要主张你给出的证明的合理性，你需要论证不存在以其他方式工作的算法。

而要证明P=NP,“只需”给出某个NP完全问题的有效算法。于是很多人会给出团问题的一个算法，然后就说他们证明了P=NP。但是一个算法本身并不构成一个证明。必须还要形式化地论证该问题的每一个实例，即在所有可能的好友关系图上，该算法都能高效地运行，并给出正确的答案。而这些证明的尝试要么没有覆盖所有可能的情况，要么本身就是错误的或不完整的。给出的算法不是结果是错误的，就是在复杂的实例上无法高效地运行。

7.4 现状

我们离证明P≠NP比以前更远了。虽然对该问题了解更深入了，但从某种程度上讲，在未来一段时间内，都不存在未被尝试过的明显的方法和已知的证明思路了。

目前唯一已知的有可能解决P/NP问题的方法，来自芝加哥大学的凯坦·马尔马利。他的工作表明通过解决代数几何学（它比高中的代数和几何要复杂得多）这一数学领域的某些问题，可能会产生P≠NP的证明。但是求解这些代数几何学问题所需的数学技术远远超出了今天人类的数学技术水平。几年前马尔马利觉得他的程序要100年才能跑完。如今他认为之前想得有点乐观了。

还要过多久我们才能看到P/NP问题真正得到解决？也许在你读到本书的时候这个问题已经被解决了。但更有可能的是，很长一段时间内它都无法被解决，也许比费马大定理从提出到解决所经历的357年更长，甚至有可能一直得不到解决，成为数学和科学领域一个永远的未解之谜。

第8章　秘密

每个人都有秘密，从密码到电子邮件，我们都有不想让别人看到的东西。P≠NP意味着某些NP问题拥有不为人知的秘密，无法很快找到它的答案。1976年，惠特菲尔德·迪菲和马丁·赫尔曼提出可以用NP问题来隐藏我们自己的秘密。从此，密码学这门研究秘密消息的学科发生了质的改变。

8.1　经典密码学简史

秘密消息的历史和人们传递消息的历史一样久远。朱利叶斯·凯撒使用过一个简单的替换加密算法，即将每个字母替换为它在字母表位置往后数第三个位置的字母。

The early bird gets the worm（早起的鸟儿有虫吃）就变成了Wkh hduob elug jhwv wkh zrup。这种字母替换的加密方法被称为凯撒密码。

加密前	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
加密后	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

图**8-1**　凯撒密码

这种技术在古罗马非常好用，加过密的消息看起来像是胡言乱语，而那时人们有限的技术还无法破译这种密码。到了公元9世纪，数学家发现了通过统计字母和短字的频率来破解密码的方法。我们来看Wkh hduob elug jhwv wkh zrup，注意其中字母h出现了4次，比其他字母都多。英语中出现频率最高的字母是e，所以可以（正确地）推测h代表e。然后注意到Wkh出现了2次，而如果h代表e，那么可以推测Wkh代表the。按此方法，很快就能破译这条消息。

在15世纪的意大利文艺复兴时期，莱昂·巴蒂斯塔·阿尔伯蒂创造了一种更为复杂的加密方法，称为多字母加密法（polyalphabetic cipher），对消息的不同部分使用不同的替换规则。这些密码在很长一段时间内都无法被破解，直到19世纪人们发展出了更系统化的密码破译方法。

作家兼诗人埃德加·爱伦·坡也对密码分析颇有研究。1839年他公开征集难以破译的密码，以供他研究密码破译的方法。一年后，爱伦·坡发表了一篇名为“谈谈秘密写作”（A Few Words on Secret Writing）的文章，他在其中声称“人类的创造力无法创造自身不能破译的密码”。他在1843年的小说《金甲虫》（*The Gold-Bug*）中就有破译密码的情节，它属于第一批描述密码的小说。

1903年亚瑟·柯南·道尔发表了《跳舞的小人》（*The Adventure of the*

Dancing Man），这本小说讲述了歇洛克·福尔摩斯如何破译一个由火柴棍小人构成的替换密码。

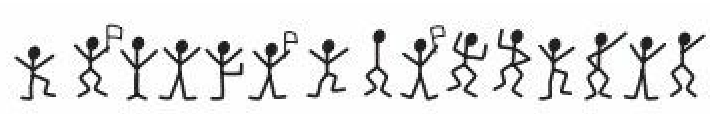


图**8-2**　跳舞的小人

随着机械时代的到来，出现了一批加密和解密消息的机器设备。其中最知名的可能要数恩尼格玛密码机（Enigma machine），是德国人亚瑟·谢尔比乌斯在1918年发明的。

恩尼格玛密码机有数个转子，在打字过程中，这些转子将字母替换成其他的字母。由于每个转子的转速不同，每一个字母具有不同的替换规则。这个机器可以看做是阿尔伯蒂的多字母加密法的一个更复杂的版本，用它加密的消息非常难以破译。



图**8-3**　恩尼格玛密码机

恩尼格玛密码机是第二次世界大战中德军使用的主要密码来源。在开战之前，波兰军事情报机构向英国提供了该机器的描述和一些破译技术。英国政府随即启动了一项代号为“超级”（Ultra）的计划来破解密码。征募的研究人员有纵横填字和国际象棋高手，也有大数学家，其中包括计算学之父阿兰·图灵。第一台可编程电子计算机“巨人”（Colossus）就是在计划的实施过程中诞生的。温斯顿·丘吉尔曾说：“是‘超级’计划让我们赢得了战争。”

密码学从诞生以来都是（而且将来也是）一场猫鼠游戏，即密码制造者和密码破译者双方的斗智斗勇。但是在20世纪70年代，随着计算机科学家开始构造基于难以求解的NP问题的加密方法，这个游戏的格局发生了重大变化。

8.2 现代密码学

“我们今天正面临着一场密码学的革命。”这是1976年斯坦福大学的惠特菲尔德·迪菲和马丁·赫尔曼所发表的著名论文中的开题句。“廉价电子硬件制造技术的进展将密码学从使用机械进行计算所带来的设计局限中解放出来，让曾经昂贵的高端加密设备变得成本低廉，使得远程提款机、计算机终端等商业化应用成为可能。”

迪菲和赫尔曼认识到了，计算机可以让复杂的加密通信协议以成本低廉的软件形态存在。但同时计算机也带来了很多挑战。随着计算机与计算机之间的网络得到普及，必须要开发出一种高效率且低成本的安全通信方法。迪菲和赫尔曼在谈到近十年来P/NP问题的进展时说道：“同时，信息论和计算机科学的理论进展很有希望提供一种安全的加密方法，从而将密码学这门古老的艺术变为一门科学。”

在迪菲和赫尔曼之前的时代，要解密一条消息必须使用加密它的密钥。如果一位将军通过恩尼格玛密码机向战场上的一位上尉发送消息，双方必须事先约定好机器的设置，即转子在消息开头的初始参数。上尉会随身携带一个密码本，里面记录了每天更换的设置参数。如果这个本子落入敌手，秘密通信就必须停止，直到发放一套新的密码本。士兵们平时会悉心保管这些本子，一旦被俘就立即销毁它们。所以并不需要经常更换新的密码本。

计算机网络带来了新的挑战，因为不能想当然地认为网络是安全的。20世纪末的计算机网络通常会在某些区域通过普通电话线来传输，而电话线是很容易被窃听的。即使是今天，坐在咖啡馆另一头的人也可以读取你使用免费Wi-Fi所发送的全部数据。

不能用网络来发送密钥，否则之后的通信变得不安全。在开始和某人秘密通信前，必须要用某种物理的方法将密钥送到那个人手上。而这样做的经济和时间成本都很高。

在前人罗杰·默克尔工作的基础上，迪菲和赫尔曼提出了解决这个问题的一种方法，他们称之为“公钥”加密。计算机生成两把钥匙，一把公钥，一把私钥。计算机自己储存好私钥，绝不把它泄露到网上。公钥则

通过网络广播给所有人。

迪菲和赫尔曼的想法是建立这样一个密码系统，即使用公钥加密信息，将原始消息变成一串密码。公钥不能用来解密消息，只有私钥可以解密消息。

假如迪菲想发送一条秘密消息“中午进攻”给赫尔曼，首先赫尔曼要生成私钥和公钥。然后赫尔曼把公钥发送给迪菲（也发送给所有可能正在监听的人），私钥则自己留着。然后迪菲用赫尔曼的公钥把消息“中午进攻”加密为密码“tzljcnnfekktis”，并把密码“tzljcnnfekktis”发送给赫尔曼。迪菲在加密时不需要知道私钥，只需使用公钥。窃听者能看到加密的字符串“tzljcnnfekktis”，但是只知道公钥是不能恢复原始消息的。赫尔曼则可以用自己的私钥把“tzljcnnfekktis”恢复成原始的消息，即“中午进攻”。

公钥加密有可能实现吗？如果P=NP那就不可能，因为会存在一个高效的算法，通过公钥来计算出私钥。

而即使是在1976年，大多数计算机科学家也认为P≠NP，在此前提下公钥加密是可能的。迪菲和赫尔曼首创了公钥加密系统，但是他们提出的通信协议不如另一个协议流行，该协议由罗纳德·李维斯特、阿迪· 沙米尔和伦纳德·阿德曼三位计算机科学家于1978年发明，并以三人的姓缩写命名为RSA。

RSA加密协议基于这样一个想法，即计算乘积容易，分解因数则很难。随机挑选两个大的质数，比如5 754 853 343和2 860 486 313，很容易计算其乘积为16 461 679 220 973 794 359。另一方面，很难逆转这个运算，即把16 461 679 220 973 794 359分解为5 754 853 343乘以2 860 486 313。RSA使用了比这大得多的数，通常有几百位长。虽然说如果不证明P≠NP，就无法证明因数分解本质上是一个难以计算的问题，但大部分人都认为因数分解是一个困难的问题。

罗纳德·李维斯特、阿迪·沙米尔和伦纳德·阿德曼因为发明该协议，分享了2002年的图灵奖。

有一个小插曲，其实RSA协议最早是在1973年被克利福德·柯克斯发明的，他为英国政府通信总部工作，该机构相当于美国的国家安全局。这

个事实直到1997年才被公之于众。

你可能每天都在用RSA加密协议。随便看一个网站（结果可能因浏览器差异而不同）。



图**8-4　Facebook**网站的地址栏

注意到https协议中的s以及后面的小锁标识。



图**8-5**　被标记的**Facebook**网站的地址栏

s代表secure（安全）。Facebook公布了一把公钥。浏览器会把用户输入的密码用公钥加密。加密过的密码被发送给Facebook。所以咖啡馆里的人即使用笔记本电脑监测所有用Wi-Fi传输的数据，也无法猜出用户的密码。Facebook用私钥可以解密用户的密码。同样地，用户的浏览器也生成一对公钥和私钥，把公钥发送给Facebook。这样Facebook就可以向用户发送经过加密的好友状态更新数据，除了用户自己别人都看不到。

8.3 P=NP下的密码学

如果我们生活在第2章描述的“美妙世界”中，密码学会发生怎样的变化？我们很容易算出5 754 853 343和2 860 486 313是否为质数，以及5 754 853 343×2 860 486 313=16 461 679 220 973 794 359。我们甚至能在几千位或几百万位的数上做这些运算。由于我们可以验证因数分解问题的解，故该问题属于NP。如果P=NP，那么因数分解问题将变得能被高效地计算，我们可以找到几百万位数的质数因子。P=NP将让RSA协议失效。所有基于公钥加密系统的协议也都将失效。P=NP意味着如果不和对方事先联系好，你将无法向任何人发送秘密消息。

难道密码学将在美妙世界中毫无用武之地吗？只有一个方法可行，就是一次性密码本（one-time pad），它的安全性是理论上可被证明的，而且不依赖于NP问题的困难程度。假设艾莉丝有一个12个字符的密码FIDDLESTICKS。所谓的密码本是一个长度相同的随机字符串JXORMQNAMRHC。从密码和字符串中分别取出第一位的字母，F和J，两者分别是字母表中第6个和第10个字母。将两者的排序相加得到16，然后用第16个字母P作为加密的第一个字母。再分别取出第二位的字母，I和X，字母表中序号分别为9和24，相加得到33。字母表中没有33个字母，于是减去26，得到7，然后用第7个字母G作为加密的第二个字母。以此类推，就可得到加密的字符串PGSVYVGUVUSV。艾莉丝把加密的字符串发送给Facebook。Facebook只须减去而不是加上密码本中的每一位，就可解密这条消息。

由于长度为12的一次性密码本和消息一样多，所有加密字符串的概率都是相等的，数学上无法通过密码本对消息本身有任何了解，无论是否P=NP，结论都成立。那为什么我们不都用一次性密码本加密，而要使用更加复杂，而且有可能被破解的因数分解加密方法呢？

使用一次性密码本必须十分小心。顾名思义，一次性密码本只能被使用一次。甚至两个不同的人分别向另外两个不同的人发送秘密消息，都最好不要用同一个密码本，否则不能保证这两条消息的安全性。每个密码本都至少要和消息一样长。没有公钥和私钥之分，只有一份共享的私钥（即密码本JXORMQNAMRHC）。Facebook和艾莉丝都必须知道密码本，但如果窃听者知道了密码本的一小部分，他就能获知部分的消息。

Facebook需要把密码本以某种别人看不到的方式交给艾莉丝（反过来也是如此）。由于人们能监听互联网通信，所以Facebook或某个受信任的第三方不得不将密码本存在U盘或者其他物理设备中然后交给艾莉丝。我能想象，美妙世界中的艾莉丝会从杂货店买一个密封的U盘，里面装满了一次性密码本。U盘会由某些受信任的组织来生产（美国政府？），并保证以安全的方式将同样的密码本发送给Facebook或其他公司。

Facebook也可以用量子力学来生成密码本并将其传输给艾莉丝。我们在下一章会讲讲量子密码学，但这个方法可能会因为成本过高而无法被广泛使用。

8.4　零知识数独

鲍勃在午餐时间尝试解开一道当天报纸上的数独谜题。

	9			8		4		
		2		4	1			5
3							6	
	1							
7	6			2			1	9
							8	
	2							8
5			2	9		3		
		4		5			2	

图**8-6**　零知识数独

鲍勃痛苦地喊道："报纸一定弄错了，这个数独不可能有解！"他的同事艾莉丝听到了，过来看了一眼。她在早班车上解出了同一道题，所以她知道鲍勃是错的。艾莉丝的解如下。

1	9	**7**	**6**	8	**5**	4	**3**	**2**
6	**8**	2	**3**	4	1	**7**	**9**	5
3	**4**	**5**	**9**	**7**	**2**	**8**	6	**1**
4	1	**8**	**5**	**6**	**9**	**2**	**7**	**3**
7	6	**3**	**8**	2	**4**	**5**	1	9
2	**5**	**9**	**7**	**1**	**3**	**6**	8	**4**
9	2	**6**	**4**	**3**	**7**	**1**	**5**	8
5	**7**	**1**	2	9	**8**	3	**4**	**6**
8	**3**	4	**1**	5	**6**	**9**	2	**7**

图**8-7**　零知识数独的解

鲍勃正打算放弃这个谜题，于是艾莉丝说她能解出这道题，但鲍勃不相信。艾莉丝知道鲍勃看到明天公布在报纸上的答案时将会很沮丧，所以她想让鲍勃继续尝试下去。她可以向鲍勃展示自己的答案，但这将毁掉鲍勃破解谜题的乐趣。艾莉丝想在不透露任何答案信息的前提下，让鲍勃相信这个数独谜题是有解的。

幸运的是艾莉丝在大学里主修计算机科学，她知道零知识证明。她设计了如下这个方案。

首先，艾莉丝回到自己的办公隔间，不让鲍勃看见她。艾莉丝随机重新排列了1～9的数字。

重排前	重排后
1	2
2	8
3	6
4	5
5	4
6	9
7	1
8	7
9	3

图**8-8**　数字

然后她把自己的解按上表进行数字替换（用2代替1，3代替9，等等），然后把结果写到一大张纸上。

2	3	1	9	7	4	5	6	8
9	7	8	6	5	2	1	3	4
6	5	4	3	1	8	7	9	2
5	2	7	4	9	3	8	1	6
1	9	6	7	8	5	4	2	3
8	4	3	1	2	6	9	7	5
3	8	9	5	6	1	2	4	7
4	1	2	8	3	7	6	5	9
7	6	5	2	4	9	3	8	1

图**8-9**　经过重新排列的零知识数独

然后艾莉丝仔细地把纸剪成81片，每片上有一个数字。她把每片纸放到一个小袋子里，并把袋子按数字原来的顺序放在一个空白的网格上。

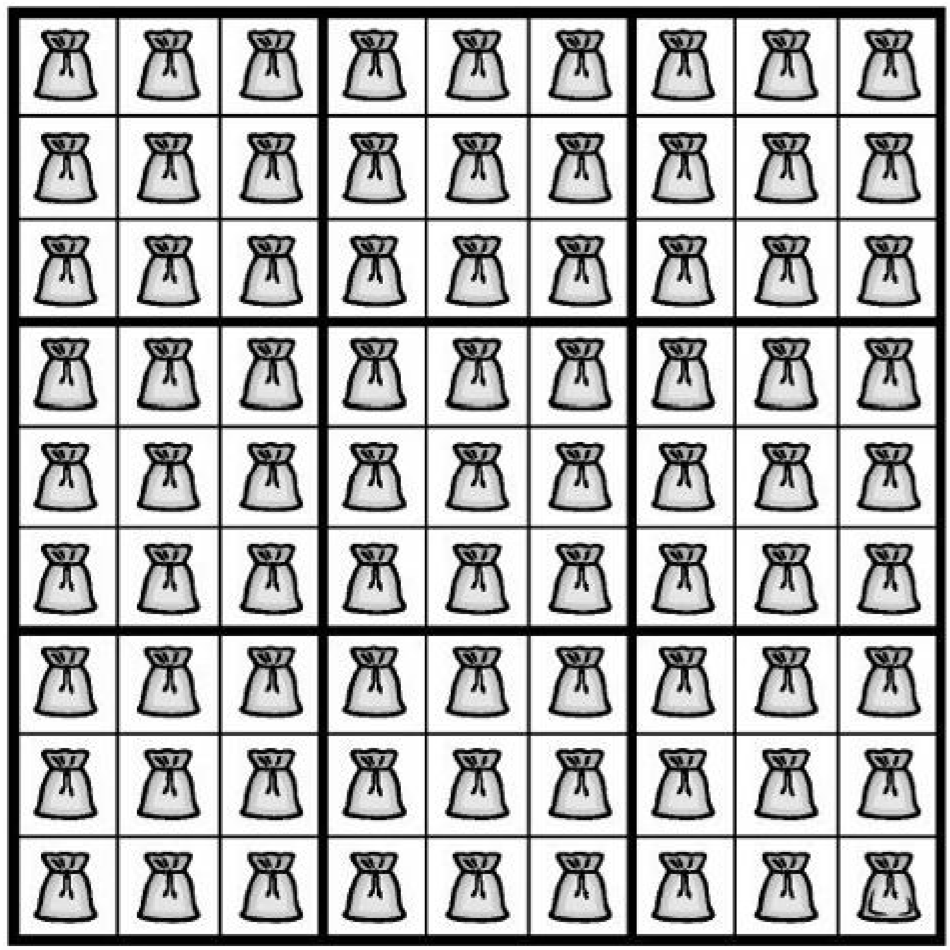


图**8-10**　被遮盖的零知识数独

最左上角的袋子里装着数字2，它右边的袋子装着数字3，等等。

艾莉丝小心地把网格和袋子带到鲍勃那里，解释了刚才她做的事情（没有透露数字重排前后的对应关系），并让鲍勃从28个检验方法中选择一种。

- 从9行中选择一行，打开那行的所有袋子。
- 从9列中选择一列，打开那列的所有袋子。
- 从9个3x3的方格中选择一个，打开里面的所有袋子。
- 选择那些位于原来谜题中已知数字位置上的袋子，并打开它们。

假如鲍勃选择了打开第3行的所有袋子。他看到的如下图所示。

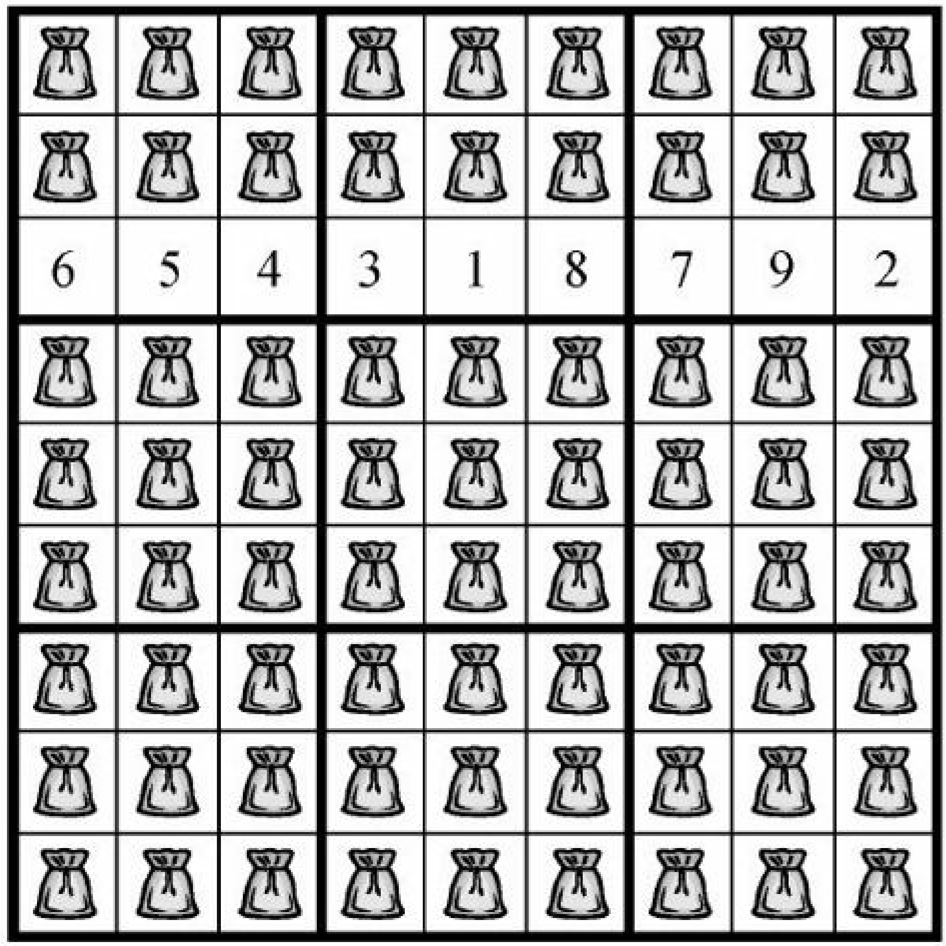


图**8-11**　打开一行的零知识数独

如果艾莉丝有一个解，并且做了她说的那些操作，那么鲍勃看到的应该是1～9这几个数字不重复的随机排列。如果鲍勃看到两个相同的数字，那他就知道艾莉丝作弊了，但是这里艾莉丝通过了测试。

鲍勃若选择列或方格也有同样的检验策略。

如果鲍勃选择最后一种检验方法，即“选择那些处在原来谜题中已知数字位置上的袋子，并打开它们”。他看到的如下图所示。

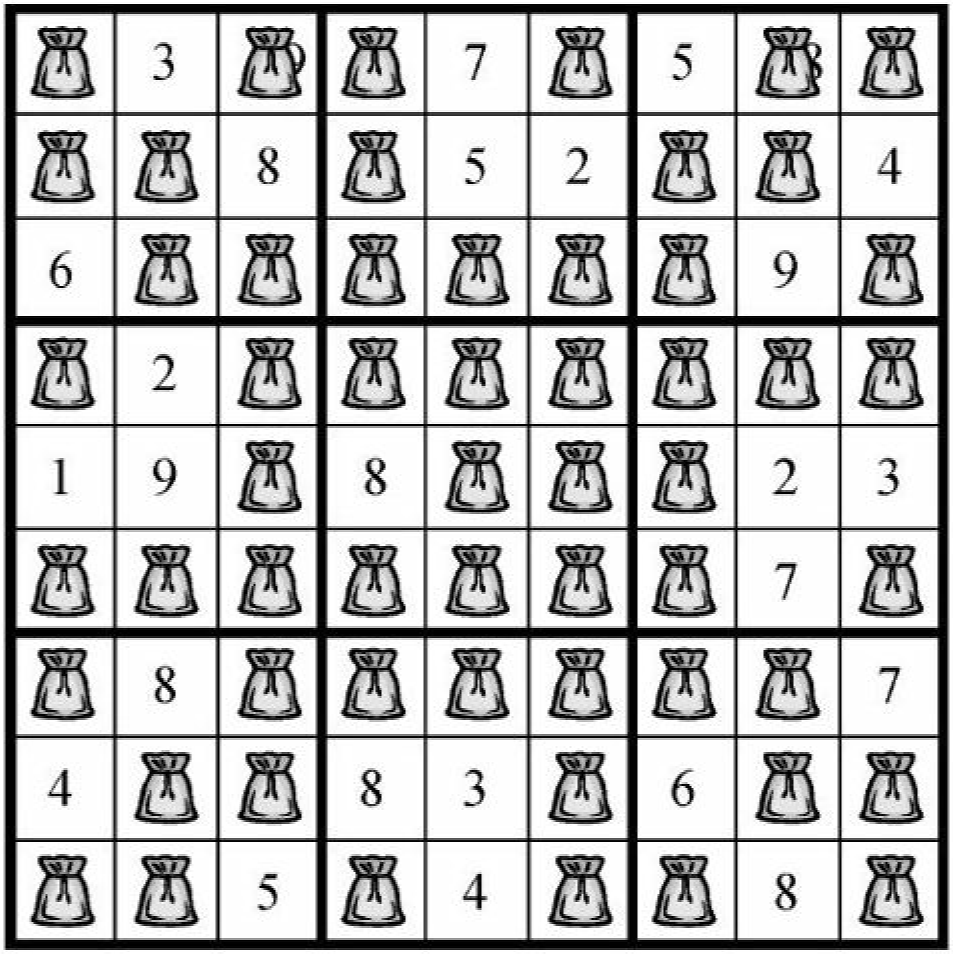


图**8-12**　打开已知数字对应位置的零知识数独

当且仅当鲍勃选择做这项测试时，艾莉丝会把数字重排前后的对应关系表展示给鲍勃看。

重排前	重排后
1	2
2	8
3	6
4	5
5	4
6	9
7	1
8	7
9	3

图**8-13**　数字

然后鲍勃就可以检查打开位置的数字与谜题中原来已知数字的对应关系是否如表中所示。根据此表，原题中的数字9应该对应袋中的数字3，果然如此。

如果艾莉丝真的有一个解并且做了她说的那些操作，她就会通过所有的测试。鲍勃从中得知了什么？如果鲍勃选择查看某一行、列或方块，他只能够看到随机排列的1～9，没有任何对解题有用的信息。

如果鲍勃选择了最后一个测试，他会看到原来谜题的一个随机重排，也

没有任何对解题有用的信息。

要是艾莉丝没有一个真正的解会怎样呢？那无论艾莉丝怎么做，都不可能通过鲍勃可能选择的所有测试。如果鲍勃随机选择的话，艾莉丝有至少1/28（即3.57%）的概率被抓到作弊。听起来这个概率挺低的，但如果艾莉丝和鲍勃重复83次测试（每次都选择一个新的数字对应关系），艾莉丝至少有1次不能通过测试的概率将超过95%。

鲍勃终于相信艾莉丝解出了这道数独谜题，但是他得到的是关于解的“零知识”，即除了知道存在一个解之外一无所知。鲍勃可以继续求解这道谜题，并相信它是有解的，但这对于找到答案没有任何帮助。

在这个例子里，鲍勃不愿意了解艾莉丝的解的任何细节。但在理论上，鲍勃可以通过抓起所有袋子并打开它们来作弊。为了避免这种情况，艾莉丝可以用上锁的箱子来代替袋子，并且只给鲍勃他选择的那些箱子的钥匙。

要是鲍勃和艾莉丝不在同一个办公室，而是位于不同的城市呢？他们仍可以轻松地通过电话或电子邮件交流，但是怎么来表示袋子呢？他们可以使用简单的加密技术。艾莉丝为每一个袋子选一个不同的随机大数字，其最后一位和袋子里的数字相同，比如为装有数字2的袋子选择3 682 502。然后艾莉丝用她的公钥为这些数字加密，把它们发送给鲍勃。鲍勃选好做哪项测试后，艾莉丝给出鲍勃选择的袋子对应的解密后的值。鲍勃可以用艾莉丝的公钥重新加密这些数字，然后检查是否与之前发送的加密值相符。

正如我们在第4章看到的，数独是NP完全问题，即每一个NP问题都能归约到数独问题。所以我们就自动有了每一个NP问题的零知识系统。艾莉丝可以说服鲍勃在敌友国存在一个很大的团，可以用三种颜色填充某张地图，或存在一条很短的旅行商路线，同时只透露团、填色方法或路线的零知识，即它们是存在的这个事实。

密码学攻击中常用的方法是通过冒充别人来蒙混过关。零知识证明则提供了一种防卫这种身份欺骗的方法。艾莉丝使用自己的公钥加密一些随机选择的个人机密信息。艾莉丝想向鲍勃证明她的身份。如果她直接把这些秘密告诉鲍勃，那么理论上鲍勃就有了冒充艾莉丝的机会。而如果艾莉丝提供的是一些零知识证明，那么她不仅能让鲍勃相信她知道这些

秘密，更重要的是没有透露自己的任何秘密。这样鲍勃就能在不知道艾莉丝个人机密信息的前提下确认对方的身份。

8.5 玩游戏

鲍勃和艾莉丝正在商量晚餐去哪里吃。鲍勃想吃牛排，而艾莉丝想吃鱼。他们决定抛硬币来选择餐馆。鲍勃抛起硬币，然后把它扣在胳膊上。艾莉丝猜正面。鲍勃拿开盖在硬币上的手，是反面，于是两人去吃牛排。

听起来挺公平，不过要是艾莉丝和鲍勃是通过电话或互联网联络怎么办？鲍勃可能会对硬币结果撒谎，甚至根本不抛硬币。艾莉丝如何才能确认过程是公平的？

一个办法是用公开的随机信息。鲍勃和艾莉丝可以商定如果当天道琼斯工业指数的收盘价最后一位是奇数，那么由鲍勃来选餐厅，如果是偶数则由艾莉丝说了算。但这个方法在星期六就失效了，股市不开盘。

这里介绍一种使用密码学的方法，用到了我们在本章前面讲到的公钥加密。首先由鲍勃生成公钥和私钥，然后随机选一个数，如69 441 251 920 931 124，把它用公钥加密。鲍勃把公钥和加密过的消息发给艾莉丝。

然后轮到艾莉丝来猜鲍勃的数字是奇数还是偶数，并把结果发给鲍勃。之后鲍勃把私钥发送给艾莉丝，后者用它解密信息，发现鲍勃选定的数字是69 441 251 920 931 124。若艾莉丝猜的是偶数，那她就赢了；如果她猜的是奇数，那鲍勃就赢了。

为什么这个方案能奏效呢？在艾莉丝选择猜奇数或偶数之前，鲍勃选好了一个数69 441 251 920 931 124。艾莉丝在猜的时候不知道鲍勃选的是什么数。艾莉丝只能看见加密过的消息，除非能破解密码，否则她对鲍勃选择的数一无所知。所以艾莉丝只能随便猜。另一方面，鲍勃无法改变选定的数，因为之前已经把它的加密版本发给艾莉丝了。艾莉丝猜完后鲍勃通过发送私钥来公布答案。如果双方都随机地进行游戏，那么胜率应该是五五平分的。只要公钥加密系统是安全的，就没有人能作弊。

抛硬币这个游戏太简单了，那么对于复杂点的游戏情况又如何呢？

鲍勃和艾莉丝能通过电话下国际象棋吗？可以，而且很容易——每个人轮流用标准国际象棋记谱法告诉对方自己的行动即可。

他们能下西洋双陆棋（backgammon）或强手棋（Monopoly）吗？鲍勃如何掷骰子才能让艾莉丝信服？他们可以采用一套掷骰子的协议，原理类似于上面的抛硬币协议。

他们可以玩扑克牌或其他牌类游戏吗？这回情况就变得复杂得多了。艾莉丝需要得到一些随机选择的牌，只有她能看见，鲍勃看不见；鲍勃也要有艾莉丝看不到的一些牌。还要有一些两个玩家都能看到的牌，以及一些两人现在看不到，但之后会让一个或两个玩家看到的牌。

好多网站都能让你玩扑克牌，但这些网站本身充当了受信任的第三方，由他们来发牌或将牌展示给各个玩家的浏览器。

在没有受信任的第三方的情况下，鲍勃和艾莉丝还能玩扑克牌吗？在20世纪70年代和80年代，密码学家设计了许多种通过电话或网络进行双人或多人牌类游戏的方案，每个玩家都有自己的私钥和公钥，并需要对加密的信息再进行一次加密。

20世纪80年代和90年代，密码学家研发出了非常通用的技术，让任意需要跟受信任对手玩的游戏都能通过互联网进行，即使没有受信任的对手也可以。这些方法既用到了加密技术，又使用了零知识证明。这些通信协议相当复杂，而且很少被实际应用。现实中人们要么依靠受信任的网站，要么会采用一些针对某个特定目的而设计的通信协议。

8.6　在云上进行加密计算

假如艾莉丝需要对敏感数据进行计算，而鲍勃是云计算服务的提供商。艾莉丝可以通过鲍勃的公钥把数据传送给鲍勃。鲍勃解密数据，做完计算，把结果用艾莉丝的公钥加密后回传给艾莉丝。如果艾莉丝和鲍勃使用的公钥加密协议足够安全，没有人可以从中得知艾莉丝的数据。如果艾莉丝信任鲍勃那没问题，但如果她对鲍勃不够信任，想让数据对鲍勃也是保密的，还能进行云计算吗？

解决这个问题需要使用所谓完全同态（homomorphic）的加密方法。RSA协议中，如果把两个数（如28和45）经过加密的版本相乘，就可得到它们的乘积1260的加密版本。不需要知道原始的数字，就能通过加密数字计算其乘积的加密版本。另一方面，RSA上的加法不满足这个条件。人们不知道有什么方法可以由28和45的加密版本得到其和73的加密版本。

大多数计算可以用数之间的和与积来表示。和类似于或门，积类似于与门，这样就能用和与积来构造电路。完全同态的加密方法允许我们直接用加密后的数字来计算加密后的和与积。使用这种方法可以从加密的输入直接得到加密的输出，不需要任何附加的通信过程。

艾莉丝可以采用完全同态的加密方案，把加密过的数据上传到鲍勃的计算机。鲍勃在执行艾莉丝指定的计算时，不需要知道艾莉丝的原始数据。鲍勃得到的计算结果是经过加密的版本，他自己无法解密这些结果，只有艾莉丝能在下载之后用自己的私钥进行解密。

很多年来，密码学家都无法实现完全同态加密，很多人曾认为它不可能实现。2009年，斯坦福大学的研究生克雷格·金特里发现了一个实现完全同态加密的方案。这个方案的效率还无法达到实用要求，但可以肯定的是，它开启了全新的可能性之门，不久的将来就会出现更好的加密协议。

8.7 创造随机性

石头剪刀布是一个很流行的小游戏，参与游戏的两个人用手势比划出剪刀、石头或布的样子。

石头能砸坏剪刀，于是出石头的人赢出剪刀的人。剪刀能剪碎布，布能包住石头。如果两人出同样的手势，那么算平手。

如何来选择玩石头剪刀布最好的策略？如果你能比较准确地预测对手，那就根据其行动来决定做出什么手势。但是让我们反过来想想，如果对手看穿了你的思维习惯，你会一直输吗？

如果你总是随机选择手势就不会。你等概率地出剪刀、石头和布，那么不论你的对手做什么，你总会有三分之一的时间赢、三分之一的时间输，以及三分之一的时间平局。话虽如此，人类很难做出完全随机的选择，这才使得在多次进行石头剪刀布的游戏时有真正的技巧性可言。

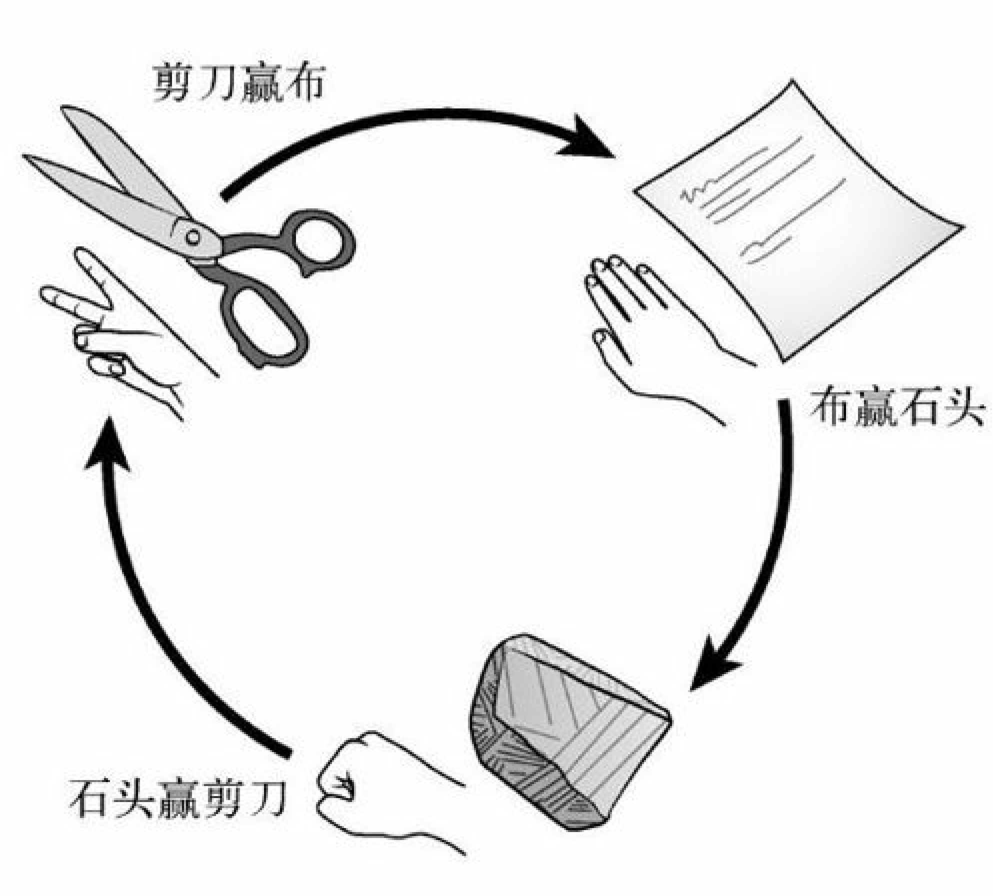


图**8-14**　石头剪刀布

类似地，密码学也要求有随机性。本章中谈到的每一个加密方法都涉及随机选择，以让信息对窃听者和对手保密。如果数字不是完全随机的，那么对手将会占优势，甚至对于一次性密码本这种安全的加密方法也是如此。

计算机如何生成随机数呢？计算机不会抛硬币，即使它会，硬币也同样遵循物理法则，不能实现真正的随机。人们也许能从量子效应中抽取随机性，但目前还不能在日常计算中实际大量使用这种方法。

人们抛硬币，掷骰子，洗牌或转动轮盘，都是为了创造随机性。所有这些操作都直接遵循物理法则，赌场完全不会有赔钱的风险。轮盘赌中小球和转盘的复杂交互作用，使得预测某次转动的结果在计算上几乎是不可能实现的，一般人几乎无法将轮盘赌结果的分布和完全的随机行为区分开来。

计算机也采用了这样的方法。它并不直接拥有完全随机的比特，而是通过执行某些结果难以预测的操作来生成“伪随机数”。密码学和随机数生成之间有直接的联系。经过加密的消息在没有密钥的对手看来应该是随机的。许多加密技术可以被改造为很好的随机数生成器。

你的计算机基本不用这些系统来生成伪随机数，而是用一些更为高效的系统，这些系统通常效果不错，但在理论上不能保证安全。如何在更少的计算时间和更高的伪随机性之间权衡是一个两难的问题。

伪随机数生成器只在问题复杂到一定程度时才有效。如果P=NP，那每一个计算上高效的过程都能在某种程度上被逆转，这样，生成计算上随机的抛硬币事件就很难了，甚至是不可能的。石头剪刀布在美妙世界将不再是一个靠运气的游戏。

8.8　持续的挑战

我们目前使用的是基于因数分解等NP问题的公钥加密系统。艾莉丝很容易就能生成一个只有她知道因子且很难被因数分解的大数，只要随机选择两个大的质数，把它们相乘就可以了。

人们还不知道因数分解是否为NP完全问题，但倾向认为不是。即使它是NP完全的，P≠NP也只是意味着存在某些难以因数分解的大数。随机选择的数字仍然有可能容易被因数分解。

现代密码学构建在NP完全问题的困难性以及P≠NP的假设之上，这是该学科目前面临的心腹之患。

20世纪70年代迪菲和赫尔曼的工作彻底改变了密码学，让我们能直接基于求解某些问题的困难程度来构造加密协议。纵横填字高手、国际象棋大师和聪明的数学家再也不能找到直接破解这些密码的方法了。

而猫鼠游戏并没有因此终止。既然不能直接地破解密码，黑客转而攻击其他的薄弱环节。艾莉丝和鲍勃可能没有在通信协议中使用很好的随机数。黑客可能会利用艾莉丝使用的浏览器或操作系统的设计缺陷来入侵她的计算机。艾莉丝可能被蒙骗，将私钥告诉其他人。艾莉丝选择的密码可能不够强壮，从而让黑客有了可乘之机。

除了通常能看到和想到的通信数据之外，黑客还能利用一些其他的信息来破解密码。比如计算花费的时间与编码的消息也许是相关的，黑客可以利用这一点。黑客还可能会破坏系统的一部分，比如把智能卡扔进微波炉加热，这样做有可能使受损的计算芯片不再保障数据的安全性。

人们也许会发明理论上牢不可破的加密方法，但基本上永远都不可能实现绝对不会泄密的保密系统。

第9章　量子

1982年，诺贝尔奖得主物理学家理查德·费曼发现，不存在简单的方法可以利用电子计算机对量子物理系统进行简单的仿真。他把这个问题转化为一个机会，提议也许可以研发一种基于量子力学的计算设备，其计算效率比传统的计算机更高。在接下来的十年，计算机科学家和物理学家经过合作，在理论上证明了量子计算机能比普通计算机更快地解决某些问题，例如因数分解。至于是否能建造出真正可用的大型或中等规模的量子计算机，并且了解这类计算机到底能做什么，不能做什么，仍然是意义非凡的重大挑战。本章我们将探讨量子计算的力量，以及与之相关的量子密码学和量子隐形传输（quantum teleportation）。

9.1 量子录像机

汤姆是波士顿人，很自然地，他也是波士顿红袜队的球迷。有一天在汤姆上班的时候，纽约洋基队客场对战波士顿，他刻意避免读到任何关于比赛的消息。下班回家后，他订了一些披萨饼，打开录像机，开始观赏几小时前就结束的这场比赛。第9局末，红袜队在二垒和三垒有人，两人出局，一人被夹杀。此时波士顿的强力击球手布莱恩·哈默走上本垒板，汤姆十分希望哈默能击中。汤姆觉得自己有点反应过度了，毕竟比赛早就结束了，哈默要么击中了，要么没击中。但是汤姆不知道是哪个结果。在汤姆看来，比赛的结果还是未被确定的，实际的结果介于输和赢之间，并且很快就会向他揭开谜底。

汤姆的真实取决于他的观测。从汤姆的观点来看，只要他没有观看到最后一刻，比赛就还没有结束，胜者还未被确定。在最后一刻之前，比赛处在一种奇怪的状态中，介于红袜队赢和输之间。

洋基队的铁杆球迷苏姗也把同一场比赛用录像机录了下来，并且和汤姆在同一时间分别观看自己的录像。苏姗和汤姆一样，不知道哈默是否能击中，如果击中那洋基队就输了。苏姗脑海中的比赛也是不确定的，是一个随机性事件，直到她看完整场比赛。

苏姗和汤姆在同一时刻、相隔200英里的两个地点观察两个随机事件。然而，他们将看到一致的结果：两个人要么都观测到哈默击打成功，要么都观测到他击打失败。不可能汤姆看到哈默击中了球而苏姗看到哈默被三振出局。两个人都不知道结果，但都确信看到的比赛结果是一样的。记录在苏姗和汤姆的录像机上的结果，以某种方式彼此纠缠在一起。

这和量子计算有什么关系？传统计算机最基本的元素是比特（bit，binary digit的简写），它只能取两个值中的一个，如赢或输、真或假。而量子计算机的基本元素叫量子比特（quantum bit）。和只能取两个值之一的比特不同，量子比特的取值能介于两个值之间。

记录在录像带上的棒球比赛并不是真的量子比特，但是它们有一些共同的属性。汤姆看球赛时，直到比赛结束，结果都是不确定的。而从他观

测到比赛结束的那一刻起，结果就确定了，红袜队不是赢了就是输了，不再介于两者之间。类似地，从量子比特被观测到的那一刻起，它就变成了一个传统的比特，只能取两个值之一，而不是介于两个值之间。

量子比特可以互相纠缠，就像汤姆和苏姗的录像带那样。让两个量子比特互相纠缠，这样它们在被观测到的那一刻就总能给出相同的答案。

不过二者的相似性就到此为止了。量子比特能以复杂得多的方式互相纠缠，而这些纠缠态可被操控，从而形成计算能力。

比赛的结果是一个简单的概率，可以在一条简单的直线上取任意值。



图**9-1**　波士顿

五角星所处的位置代表波士顿有30%的概率会赢。汤姆看完了比赛，五角星要么左移到尽头，要么右移到尽头，至于向哪边移动则取决于比赛结果。

量子比特则可在一个圆圈上取值。



图**9-2**　量子比特

这张图里的五角星在二维平面中的坐标是(0.55真, 0.84假)。量子比特也可以取负值。笑脸代表的量子比特取值为(－0.71真, －0.71假)。量子计算机能以可控的方式旋转或翻转这个圆圈。

一个量子比特可用一个二维的圆圈来表示。两个量子比特则需要用一个四维版本的圆圈来表示其状态，很难在这里画出来，甚至难以想象。30个量子比特则需要超过1万亿个维度。

这启发人们可以用量子计算机的方法解决NP问题。想想从2万敌友国居民中找出50人的团的问题。我们能用大概500个量子比特表示出所有可能的50人的集合。然后可以同时并行地处理所有的集合，并通过适当的旋转和翻转操作标记出其中能够组成团的那些集合。

我们现在有了一个“量子态”，即大概3×10^{150}（3后面跟150个0）组敌友国居民的组合状态，其中某些被标记为团。如果能用某种方式很快地把这些团从量子态中抽取出来，我们就能用量子方法快速解决团问题以及每一个其他的NP问题。当我们观测这个量子状态的时候（就好比看到录像带上比赛的结尾），我们只能看到一个结果，即敌友国中的一群人，而很有可能这群人不会组成一个团。

我们需要某种方式，让组成团的人群集合脱颖而出，从而更有可能被观测到。这可以用量子操作来实现。最笨的方法将使用和集合的个数相等次数的量子操作，即大概3×10^{150}次的量子操作，完全没有体现使用量子计算的优势。1996年，在新泽西州贝尔实验室工作的洛夫·格罗佛发明了一种更高明的量子算法，“仅仅”使用2×10^{75}次的量子操作。即使我们每秒做1万亿次量子操作，所需时间也大致是当前宇宙年龄的5倍。

有某些证据表明格罗佛的算法是使用量子计算机来求解NP完全问题所能达到的最好结果，所以P=NP的量子版本也不太可能成立。即使物理学家们成功造出了量子计算机，它还是无法解决我们面临的最难的问题。

这并不意味着量子计算机就没有用了。它能对纳米级别的物理系统有效地进行复杂的仿真，这将有助于揭开宇宙的一些未解之谜。量子计算机还能求解某些人们无法用传统计算机有效解决的NP问题。

1994年，贝尔实验室的另一名研究人员彼得·肖尔意识到，量子计算机能对数字进行因数分解，例如对于数字16 461 679 220 973 794 359，找到两个数5 754 853 343和2 860 486 313，并有5 754 853 343×2 860 486 313=16 461 679 220 973 794 359。他的算法工作在快速的量子计算机

上，对几百位甚至几千位的大数都有效。搜索一个数的因子问题有着良好的代数结构，可被量子计算机利用。尽管我们认为因数分解对于今天的计算机是很难的问题，量子计算机则能够借此克服这些困难，分解很大的数字。而NP完全问题则缺乏这种良好的代数结构，故肖尔的算法对通用的NP问题是无效的。

当然，我们能真正使用格罗佛或肖尔的算法的前提是：有一个能工作的量子计算机。为解决今天的机器无法解决的规模可观的问题，我们至少需要上万个量子比特彼此纠缠，并在几秒的时间内是可控的。可惜的是，量子纠缠态十分脆弱。量子系统和外界环境的任何交互作用都可能引起一次“观测”，导致某些纠缠态的丧失，这对于精密的量子计算是不可挽回的灾难。

甚至对于两个量子比特，目前物理学家还不能使其达到完美或接近完美的量子纠缠态。计算机科学家采用所谓量子纠错的方法来设计算法，以处理中等规模的量子纠缠。即使如此，我们不知道如何在超过5个量子比特之间创造出显著数量的纠缠态。可能存在某些自然界的基本法则，阻止量子之间在一段足够长的时间内处于显著的纠缠态，也可能这只是一个棘手的工程问题。我们还是让物理学家来解决这个问题吧。

还有其他利用量子效应来进行计算的方法，如量子绝热系统，或量子退火，而它们也分别有自身的技术和计算局限性。一个叫D-Wave的公司宣称建造了基于这些技术的机器，但这些机器的计算能力能否超过人们的桌面电脑，还有待评估。

即使我们发现了建造真正的量子计算机的方法，这些机器仍然是为特殊用途而建造的，如因数分解或者量子系统的仿真。它们也许有助于破解密码，以及加深人们对宇宙的本质的了解，但不太可能用它们来求解NP完全问题，或加快电子表格的运行速度。

9.2 量子密码学

我们第8章中讨论的大部分密码学工具的安全性都依赖于一个假设，即因数分解是一个计算上困难的问题。任何人只要口袋里装着量子计算机，就能用肖尔的算法把密钥分解因数进而破解这些密码。虽然现在还没有装在口袋里的量子计算机，但将来有可能出现。密码学家可以通过设计基于其他困难问题的通信协议来设法绕过量子攻击，而这些问题必须缺乏量子计算机能利用的良好代数结构。量子学术界则为我们提供了另一个解决方案，即基于量子力学的密码学。

我们对于复制计算机数据这件事已经习以为常了。差不多每一个计算平台都会提供复制和粘贴功能。我们可以把文件的多个副本放在不同目录下或不同的机器上。我们能把数据备份到硬盘或是“云端”。这带来的麻烦是，信息的副本太多了，以至于很难彻底删除某些个人文件或电子邮件，并确保其他的地方不留下相应的副本。

与之相反，量子比特不能被复制。复制一个量子比特就需要观察它，而无论以何种受限的方式来观察，都将导致量子比特转变成普通的比特。乔治想把一个量子比特递送给哈利。如果埃里克在途中试图复制或读取，那个比特将变成一个普通的比特。当然这本身并不构成一个发送秘密消息的方式，毕竟消息接收者哈利也必须要读取这个比特，并导致它的性质改变。

1979年，来自蒙特利尔大学的吉尔斯·布拉萨德在波多黎各参加了IEEE计算机科学基础会议（Foundations of Computer Science conference），该会议是理论计算机科学界的主要会议之一。IBM的查理·贝内特找到了在海滩游泳的吉尔斯，这次海边相会开启了两位科学家一段令人惊叹的科研合作，其工作成果包括使用量子比特来创造理论上不可破译的密码的方法。贝内特和布拉萨德发明的方法通过混合使用经典和量子加密技术，既能让乔治和哈利成功地传输一把密钥以供将来加密消息，又能侦测到该密钥是否已被监听者截获，那样的话他们可以再试一次。

这个通信协议也有局限性。为处理可能出现的小错误，不仅要增加所需的量子比特的个数，也让监听者埃丝特有机可乘，有可能不被察觉地做些手脚。人们又研发了一系列更复杂的通信协议来应对这些新问题。

和量子计算不一样，贝内特–布拉萨德协议不需要量子达到彼此纠缠的状态，而且已经实现了可以实际运转的中等规模的量子加密系统。洛斯阿拉莫斯国家实验室的研究者已成功地通过148千米长的光缆发送消息。还有人通过大气在加那利群岛的两个岛屿间发送了信息，距离也将近145千米。在不远的将来，我们将能用卫星来发送使用量子技术的不可破译的密码。

那我们为什么不为所有的秘密通信需求应用量子密码学呢？量子密码技术仍处于实验期，十分昂贵、带宽有限而且容易出错。密码系统的漏洞通常不在于使用的加密方案能够被破解，而在于实际的执行操作中的薄弱环节。如果量子通信协议投入实际应用，其操作安全性很可能和经典方法一样差，甚至有可能更差。另外，目前没有已知的方法能让量子密码通过互联网传输，因为信息通过互联网从源头传输到目的地的过程中要经过多次路由的转发。最重要的是，我们很可能无法在未来较短时期内看到能解决因数分解问题的计算机，无论是量子计算机或是其他的计算机。目前所用的密码在很长一段时期内都将是安全的。

9.3　量子隐形传输

1996年2月的《科学美国人》杂志在封二刊登了IBM的广告，宣传该公司的研究新进展。

> 很多年来她和住在大阪的朋友分享食谱。她向他展示了几百种使用红辣椒的方式。他则分享了寿喜烧的独家秘方。有一天，玛格丽特女士给小泽先生发了一封电邮，里面说：“做好准备，我给你心灵传输（teleport）了一份炖牛肉。”玛格丽特女士对技术的要求稍微有点超前，但我们一直在为实现她的愿望而努力。来自IBM的科学家和他的同事们发现了一种方法，能将一个物体原地分解、再在另一个地点毫发无伤地重现它。听起来像变魔术，但是他们取得的突破性进展将影响到科技生活的方方面面，从研发未来的计算机，到拓展人类对宇宙的认知。真是一群聪明的家伙！但他们中还没有一个人能把肉馅塞进卷心菜里。



图**9-3**　**IBM**的广告。由**IBM**公司提供

广告中标榜的技术是怎么回事？与其说它和炖牛肉有关系，不如说和量子比特有关系。而量子比特真的可以原地分解，再在另一个地点重现，就好像变魔术一样。广告中提到的研究人员就是1993年在IBM工作的科学家查理·贝内特和他的同事们，其中包括吉尔斯·布拉萨德。为我们带来量子加密技术的这群人，也发明了量子隐形传输。

假如阿瑟想把一个量子比特传输给哈瑞特，该如何做呢？

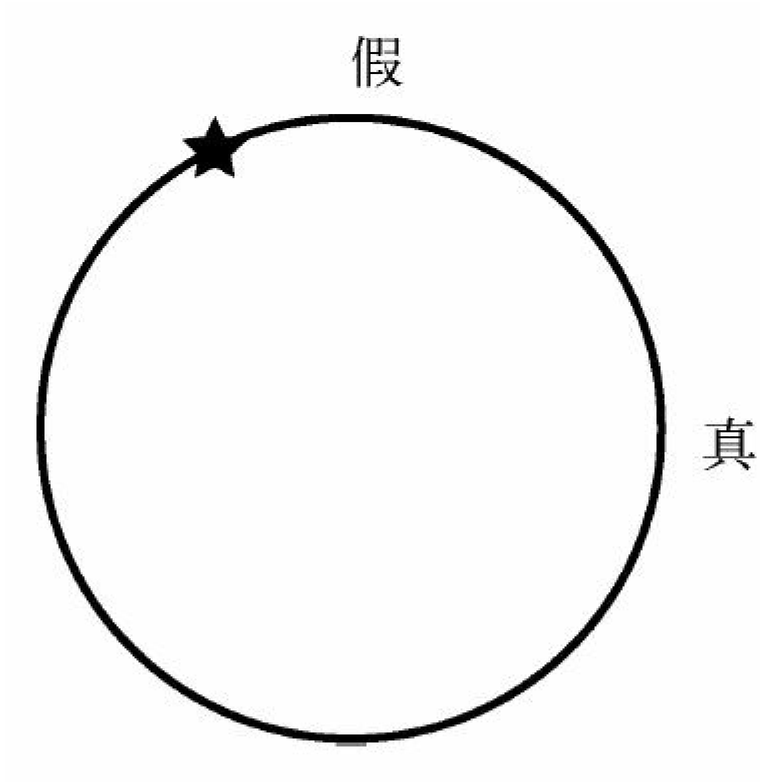


图**9-4**　量子比特

阿瑟可以选择用联邦快递来寄送这个量子比特，但是如果海关人员想查看包裹内容，就会毁掉这个量子比特，让它变成只能取真假二值之一的普通比特。即使阿瑟十分小心地亲自运送这个量子比特，他也不可能完全避免它在与外界环境的相互作用中被损坏。

阿瑟也可以发送这个量子比特的描述信息(-0.55真, 0.84假)，哈瑞特根据描述重建一个完全相同的量子比特。但如果阿瑟只有量子比特本身，他无法对其准确的描述做出判断，因为任何试图测度量子比特的行为都将导致它坍缩为真或假二值之一。

贝内特和同事们为阿瑟向哈瑞特传输量子比特提供了一种方法，但有一个缺点。阿瑟和哈瑞特需要事先准备一些彼此纠缠的量子比特。

图中带心形的量子比特处于纠缠状态，分别为阿瑟和哈瑞特持有。如果阿瑟观测他的量子比特，他会等概率地看到“真”或“假”。但若阿瑟看到

的是“真”，那哈瑞特看到的必然是“真”，如果他看到了“假”，那她看到的也是“假”，这和本章前面介绍的棒球比赛是类似的。

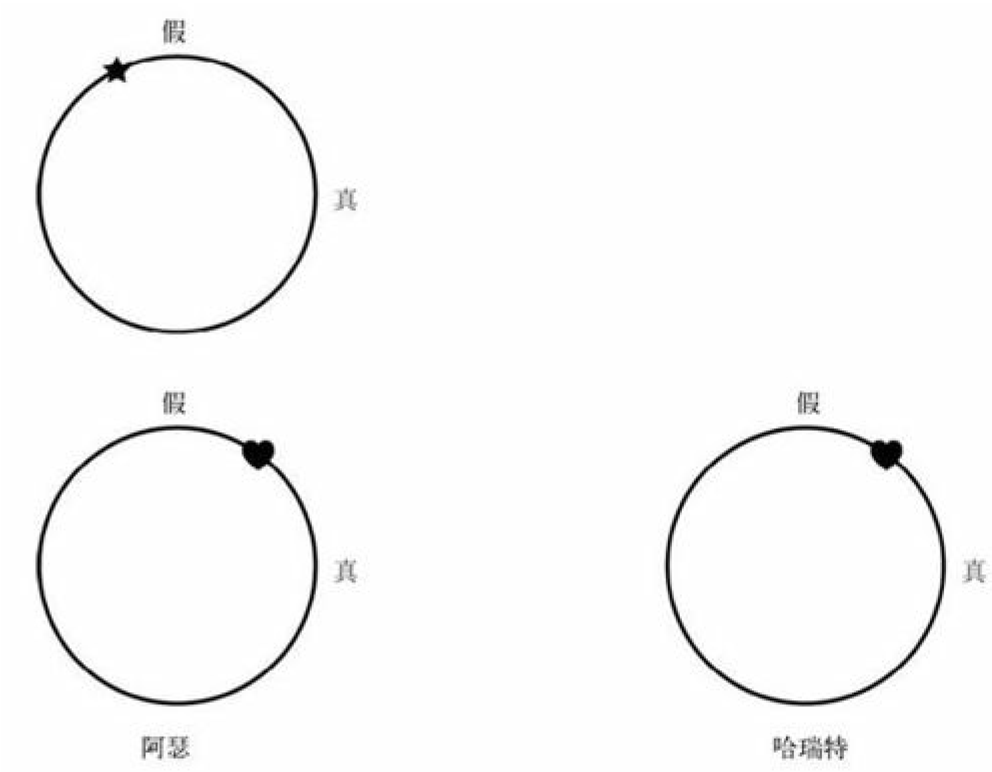


图**9-5**　纠缠

阿瑟现在有两个量子比特，标有五角星的用来传输，标有心形的和哈瑞特的量子比特处于纠缠态。之前提到过，这两个量子的组合态在四维空间表现为一个球体。阿瑟在不观察量子比特的情况下小心地旋转这个四维球。然后阿瑟观测这两个量子比特，得到两个普通的比特（0或1）。

阿瑟把这两个普通的比特传送给哈瑞特。其中一个比特告诉哈瑞特是否要旋转她的量子比特，另一个则表示是否要翻转它。哈瑞特根据指令进行相应的操作。接着轻吹一口气，大喊一声“变！”袖子里什么都没藏！突然，哈瑞特就得到了阿瑟原来的那个量子比特。令人震惊的是，这一切都是真的。

阿瑟的两个量子比特发生了什么？它们在被观测到的那一刻就毁灭了。否则，阿瑟将制造出量子比特的副本，这是量子力学法则所禁止的。

当阿瑟操作和测量他的量子比特时，哈瑞特的比特未受影响。阿瑟不能通过对自己的量子比特的操作而影响到哈瑞特的量子比特，否则阿瑟就有了和哈瑞特通信的方法，这种信息传递的速度比光速还快，这在物理学上是不可能的。

当哈瑞特接收到阿瑟传来的两个比特时，这个信息能让她把自己的量子比特看做阿瑟原来的量子比特经过量子操作后的版本。这两个普通比特准确地告诉了哈瑞特要如何逆转这些操作，才能恢复阿瑟原来的那个量子比特。哈瑞特无法检测阿瑟的操作，但随着阿瑟传送过来的两个普通比特，一个完好无损的量子比特就像变魔术似的被传输过来了。

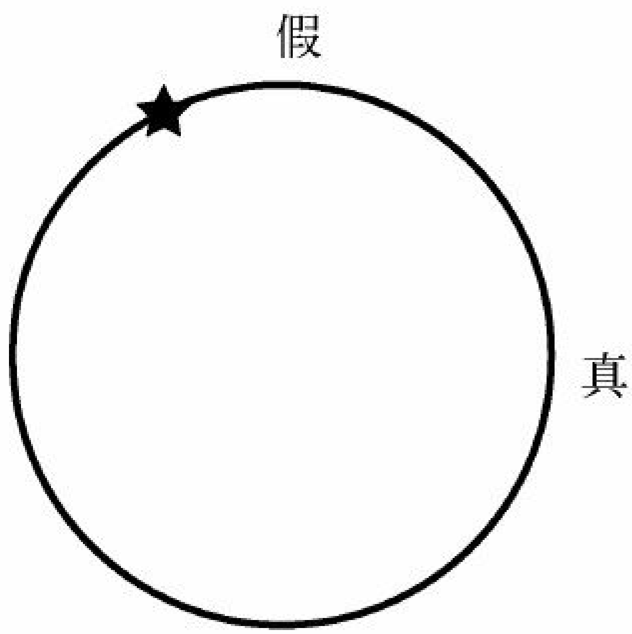


图**9-6**

怎么把这一切和炖牛肉或者《星际迷航》中那些厉害的传送器联系在一起？假如我想用量子隐形传输从芝加哥到东京。某个公司会为我准备一大堆处于纠缠态的量子，把其中的一半小心地运送到芝加哥，另一半运送到东京。然后我在芝加哥的一个小房间里和数量多到足够描述我的量子比特待在一起，在某个高维空间被旋转，然后经过测量变成经典的比特。这些比特被传送到该公司的东京分支机构，然后那里的人们在一个小房间里以恰当的方式旋转另一些量子比特，打开房门，我就出现了。酷！

那堆彼此纠缠的量子就这样被消耗了。要传送别人或者把我传回去就需要一堆新的处于纠缠态的量子比特。其实道理和航空燃油差不多。我从东京坐飞机回家，不可能重新使用那些在来东京的航班上消耗的燃油。

更令人担心的问题是如何让那些描述我的海量量子（规模可能达到3个10亿相乘的数量级，基本上相当于构成人体的原子数量）保持在完全的纠缠状态，这可能还严重低估了所需的量子比特数量。即使有极小部分的量子和外界环境发生轻微作用而丧失了纠缠态，从另一头出现的我就很可能被毁形，甚至变成一团死肉。所以你先走，我宁愿慢一点，消耗一些航空燃油，咱们东京见。

9.4　量子的未来

有些研究者认为应该从量子角度重新审视所有的计算问题。即使量子计算机不能解决NP完全问题，它对量子世界的仿真也将极大促进人类对物质、宇宙乃至人类大脑的认知，从而推动科技的进步，取得今天所无法想象的成就。

还有人看到，量子研究从20世纪90年代中期开始在算法和硬件两个方面的进展都十分缓慢，据此怀疑能否看到量子在计算设备中起到关键性作用的那一天。如果缺乏革命性的进展，量子计算技术在未来一段时期内仍将停留在科幻作品中。

如果量子不是计算领域的下一个浪潮，那什么是呢？有哪些巨大的计算挑战在前方等待着我们？且待下章分解。

第10章　未来

我本人对P/NP问题得到解决的前景持悲观态度：我认为P≠NP，而且此生都看不到它的证明。我们不会见证第2章中美妙世界的到来，但是也不能排除其可能性。我认为P/NP问题在未来的几个世纪内仍将是一个未解之谜。

P/NP问题不仅仅是一个数学上的异类。虽然我们不能直接解决它，但研究它的过程赋予了我们一种通用的框架，有助于思考如何应对从实际需求中产生的那些困难的问题。今天人们在计算学方面面临哪些重大挑战？

- 并行计算：曾经每过18到24个月，计算机的计算速度就提高一倍，但现在我们正在触及物理极限，将来很难制造出比现在快得多的处理器。另一方面，计算机正在横向扩张，可以让多个处理器一起工作，不论是在同一块芯片上，还是在云端。如何调整我们的算法，以适应这个日趋并行化的世界呢？

- 大数据：从互联网到科学实验再到仿真研究，我们每天都在生成海量的数据。如何试图理解、感受、学习如此庞杂的信息，并形成预测能力？

- 一切事物的网络化：世界上大部分地区都有计算机网络，无论人们使用的是像Facebook这样的社交网络，或者仅仅通过电子邮件交流。很快几乎所有人造的物品都将成为这张大网的一部分，从我们穿的衣服，到阅读时提供照明的灯泡。我们如何更好地利用这个超级互连的世界？

无论最终的答案是P=NP还是P≠NP，对这一问题的研究过程本身都将在很大程度上影响我们应对这些挑战的方式。

10.1　并行计算

1965年，戈登·摩尔注意到一种称为晶体管的基本元件在单块计算机芯片上的数量每年都在急剧增加。摩尔大胆预言，单块芯片上的晶体管数量大概每两年就会翻一倍，这一趋势将持续至少10年的时间。现在人们称其为“摩尔定律”，这个定律从1975年直到今天都有效，甚至在今后的很多年内都将有效。

在很长一段时间里，摩尔定律意味着计算速度的飞快提升。2005年前后，计算机开始触及物理极限，即让处理器跑得更快一点所消耗的能量超过了提升的速度能够带来的好处。人们反而要稍微降低处理器的速度，以改进能源的消耗。

然而我们还是往芯片里塞更多的晶体管。要这么多晶体管做什么用呢？现在这些芯片在同一时间可以做几件事情，以并行的方式工作，从而比同一时间只做一件事的方式更快地解决问题。

让我们来看一台特殊的机器：IBM公司的华生（Watson）。2011年2月，它在竞猜节目*Jeopardy*的第一集中得了冠军。华生由90台IBM POWER 750服务器组成，每台都有4个POWER7处理器。一个POWER7处理器的内部其实有8个处理器（也叫核心），所以它能同时做8套运算。这样，每台服务器就有32个核心，而华生系统一共有2880个核心。华生能同时做2880套计算，所以它能理解*Jeopardy*中给出的“答案”，并判断每一环节中是否该抢先对手按下抢答器。根据摩尔定律，在不远的将来，2880个并行处理核心显得微不足道，几十年后我们可能有百万核或几十亿核的计算机。

并行技术的发展对很多计算机科学领域提出了新的技术要求。一台计算机如何决定怎样把计算量在数个核心或是数台电脑上分布开来，以达到最佳的效能？我们需要考虑多核计算机因素并重写我们的通用编程语言吗？如果需要，该如何做呢？

P/NP问题同样受到并行技术的影响。记得本书开篇提到的《查理和巧克力工厂》中的小女孩维鲁卡·索尔特吗？她想要找到藏在巧克力包装纸里的金券。她爸爸就采用了并行技术，把大量的巧克力分配给几百个剥

花生壳的工人，让她们同时来找金券。人们也可以这样来求解NP问题，把所有可能是团的集合分配给几台计算机以及每台计算机的几个核心。这可能会让一个持续数周的计算在几小时内结束，但对于某些NP问题的求解还是没什么帮助。如果我们动用100万台计算机，每台有10亿个核心，每个核心每秒能进行1×10^{18}个操作，仍然需要花费几乎10^{100}倍于宇宙寿命的时间。P/NP问题在并行的世界里仍然是有效的。

而那些P类的问题，即我们能高效地解决的问题又怎么样呢？我们能最大限度地利用多计算机和多处理器的优势吗？大多数情况下我们可以通过修改算法来实现这一点。从基本的算术到找最短路径再到匹配问题，所有这些问题都有能扩展到多核来并行计算的算法。

像P和NP一样，对于那些能用并行算法快速解决的问题集合我们也起了一个名字，叫做NC，代表Nick's Class，它得名于并行算法的先驱尼古拉斯·皮彭格。如果P=NC，那任何一个我们能高效解决的问题，都能通过并行计算更快地解决。我们不知道是否有P=NC，更不知道是否有NP=NC。NP=NC意味着每一个NP搜索问题都能用极快的速度在并行计算的计算机和（或）核心上找到有效的解，这预示着一个比P=NP更加美妙的世界。NP=NC的可能性很小，但它和P=NP一样，也是一个巨大的未解之谜。

10.2　处理大数据

在1秒钟的时间里，人们生成了35分钟的YouTube视频、1600条Twitter上的消息、11 000条Facebook消息、50 000次Google搜索请求，以及300万封电子邮件（不过其中90%都是垃圾邮件）。

哈勃望远镜从它的轨道扫描太空，每秒传回20万字节的消息。1字节数据大概就是1个字母。哈勃望远镜的继任者詹姆斯·韦伯太空望远镜拥有更大的抛物线形的镜片，每秒将向地球传回350万字节的数据。

坐落在法国和瑞士边境线上的大型强子对撞机（LHC）是世界上最大的粒子加速器。它平均每秒生成5亿字节的数据，每一年的每一秒都是如此，而一年有3100万秒。

CERN即欧洲核子研究组织，它是大型强子对撞机的建造者和维护机构。那里的科学家们还建造了一个LHC计算网格，将产生的大量数据分发到位于34个国家的服务器上，让全世界的科学家都能访问并分析LHC的数据。

人类DNA序列包含大概5500万字节的信息。全世界70亿人的基因序列大概包含4×10^{17}比特的信息。这还只是人类一个物种。

我们很容易就能制造各种廉价的传感器，测量从温度到动作、从声音到辐射的各种事物。每一个传感器都在持续生成信息，而实际中通常会使用上千个传感器。有人形容美国军队“在传感器里游泳，在数据里淹死”。

不是所有数据都来自外部世界。许多科学实验实际操作起来十分困难或者成本昂贵，于是人们借助计算机来进行仿真实验，也可以了解生物学、物理学和化学系统中的各种交互作用。这些仿真实验同样生成了大量的有待分析的数据。

上述各种来源的数据虽然数量很大，但其中大部分都是无用信息，有的是随机噪声，还有的是冗余数据。找到数据集中关键而有用的部分是很困难的，并且需要对信息的深度解读。如果P=NP，我们只要用算法把

数据过一遍，就能挑出其中重要的部分，再使用基于奥卡姆剃刀法则的工具，就能形成对数据的理解和预测能力。

而我们不太可能生活在那样一个美妙的世界，我们只能尽量根据实际需求寻找和修改我们的算法。从大量的数据中理出头绪是一个非常困难、但也十分重要的计算问题。

大数据往往是好事，尤其是在机器学习领域。机器学习研究的是如何在给定的数据集上训练算法。拥有的数据越多，训练出的算法就越好。通常，拥有更多数据胜过拥有更好的算法。Google在垃圾邮件检测、语音识别和语言翻译等方面能有不错的表现，靠的就是它所能使用的大量样例数据。

在不远的将来，我们拥有的数据将帮助我们更好地分析个人健康状况，建造更智能和节能的电力网络，也将让自动驾驶的汽车得到普及，并让我们对所处的宇宙的基本性质有更深的了解。如何理解数据以更好地丰富人们的生活，是计算机科学家面临的一大挑战。

10.3　一切事物的网络化

全世界大概有20亿人通过互联网彼此联系，如电子邮件或社交网络。如今互联网带给我们的交流、合作、学习和休闲方式，是在20世纪不可想象的。

如果我们把各种物品也放到互联网上会发生什么？在不久的将来，我们将拥有各种能上网的小型廉价芯片，无论是通过Wi-Fi、移动蜂窝网络，还是通过其他正在研发的无线通信系统。我们能把这些芯片放到几乎任何东西里，从我们穿的衣服到我们开的车，再到我们买的食品的包装。你不用纸笔做记录就能追踪一切，从孩子们是否系好安全带到你每天卡路里的摄入量。没有牛奶或洗发液了？别担心，补给品已经自动发货，正在到你家的路上。你再也不会误服药品了。你穿的衣服会告诉你它们是否适合特定的场合和天气，或者会说："你真的想穿这件衬衫来搭配这条裤子吗？"这对于那些色盲人士或者不懂服装搭配的人一定大有裨益。

你再也不会丢钥匙、钱包、门票，或者任何其他的东西了。你甚至都不用携带这些东西。手机发一个信号就能让你家门锁在你接近时打开。自动取款机直接会把钱交到你手上。你可以从商店里拿起东西就走。所有的付款和扣费都是自动发生的。

上述场景的实现需要所有的设备都能彼此协作，并拥有良好的配合意识。需要研发出一种合理的方式，让这些设备能够交换信息，同时保护个人的隐私。我们将在前所未有的程度上实现协同。想象如果所有的汽车能够彼此合作，交通该有多顺畅。而这些进步的实现，都要求算法能对某些大规模问题进行快速、稳定和持续的求解。除非交通系统能够快速响应，否则一次很小的交通事故也可能造成大面积的交通瘫痪。这些问题中一部分涉及了P/NP问题，而我们只能尽最大努力去处理好它们。

Sun公司（现在是甲骨文公司的一部分）在20世纪90年代有一句口号："网络即计算机。"网络上彼此独立工作的机器通过协作，在整体上可被看做一个计算设备。当一切事物都能被网络化，我们就有了一个非常大的计算机。驯服这头猛兽虽然不易，但它也将为我们撞开一扇充满新的机遇的大门。

10.4　应对科技变革

并行计算、大数据和一切事物的网络化并不只是出现在科幻作品中的概念。它们更是一场早已开始的变革的宣言，而这场变革将在未来的一二十年里横扫世界。社会将会发生怎样的变化？

几乎不可能事先预测某项科技创新对于社会的巨大影响力。在20世纪50年代，经营着一家卡车公司的马尔科姆·麦克莱恩发明了一种不用卸货就能直接从轮船移动到卡车上的集装箱。他的发明推动了运输业的一场革命，在这场革命中诞生了可容纳上千个集装箱的巨型轮船，也孕育了不需要任何码头工人就能运转的自动化港口。这个历史事件在当时的文化产物中也有所体现，如电影《码头风云》（*On the Waterfront*）和电视剧《一家子》（*All in the Family*）。这跟中国的崛起也有一定的关系，借助一个个钢制的箱子，中国将出产的货物运送到了世界各地。

汽车推动了郊区的崛起。手机意味着我们再也不用提前做好计划。Twitter帮助人民推翻了政府的统治。科技对人类的影响体现在各种难以预料的方面。人们最好做好准备，因为这一切马上就要再次发生。

除了要考虑计算的问题，我们还需要考虑人的因素。人们总会利用新的科技做一些坏事，有的人是无意的，另一些人则是故意为之。我们还需要具备符合人们认知逻辑的良好产品设计，让人们能轻松地上手使用新一代的科技设备，尽可能不要让学习曲线太陡。

可能会有人尝试控制科技，其动机有可能是出于个人利益，还有可能是为了获取他人的信息。最坏的可能性是人们会用科技制造麻烦。有些麻烦无伤大雅，只是比较烦人而已，另一些则会造成大量的财产损失，甚至造成人员伤亡。为了阻止某些人利用科技手段制造社会灾难，人们有必要加强密码学和安保措施的研究和使用，并且时刻保持警惕。

需要注意的是，当一项科技在大部分情况下都表现良好时，大家会误以为它能在所有情况下都表现良好，这也许是其最危险的一面。如果人们对科技投入过多的信任，它可能就会在大家毫无防备的时候突然掉链子。未曾预料到的罕见事件的发生会加剧这个问题的严重程度。一些例子包括：卡特里娜飓风袭击下新奥尔良市的防洪堤决口，2010年深水地

平线（Deepwater Horizon）公司的钻井平台爆炸之后大量原油泄漏的事故，以及2011年日本地震和海啸之后福岛核电站核泄漏灾难。我们应该像对待未被驯服的猛兽那样正视科技的力量与危险性。发生大规模科技事故的概率虽然很小，但仍有可能性。人们要警惕不要让科技事故酝酿出更加巨大的灾难。

10.5 关于P/NP问题的结束语

证明P≠NP并非易事。你需要证明不存在有效的算法能解决团问题或任何其他的NP完全问题，这些算法既包括现有的也包括将来发明的。如何证明每一个潜在的算法都必将失败？

人们相信这样的证明总有一天会出现，但有可能需要过上20年，200年，甚至2000年。总有一天，当人们发展出的新技术最终能证实P≠NP的时候，数学家们会兴高采烈，为一个伟大的问题得到一个伟大的解答而欢呼。而求解P/NP问题过程中产生的技术将让我们深刻认识高效能计算的威力，这种东西会渗透到社会生活的方方面面。

P/NP问题远远不只是一个数学谜题那么简单。它是一种思考的方法，一种根据问题的内在难度对其进行分类和认识的方法。虽然尚没有P≠NP的确凿证据，我们起码知道了：当面临一个NP完全问题时，不可能找到一个在所有情况下都能解决该问题的算法。这时就需要借助于其他的工具，如近似计算、启发式方法、暴力破解等方法的组合，然后尽我们所能地争取最好的结果。NP完全性理论给了我们一个通用的思考框架，允许我们建立一套由很多技术组成的工具体系，向那些难以计算的问题发起有效的攻击。

P/NP问题让学术界团结在一起。物理学、生物学、经济学以及其他许多领域中都有NP完全问题的身影。虽然物理学家和经济学家所关注的问题有很大的差异，但这两类问题也存在着共性。所以分享各自的工具和技术将为双方都带来巨大的好处。例如，物理学中为了寻找物理系统的基态而开发的工具，对于经济学中寻找复杂经济环境中可能出现的均衡行为也是有帮助的。

P/NP问题内在的难度同样促进了新技术的发展。当代密码学家从P/NP问题受到启发，把密码学操作过程从一门艺术变成了科学。对于解决P/NP问题的强烈需求也在激励人们建造更快、更强的计算系统，催生了诸如量子计算这样的新技术。

计算是一种和过程有关的活动，而过程不仅仅出现在计算机上。P/NP问题归根到底与自然本身的极限有关，与生物和物理系统进化的极限有

关，甚至可以说与人类思想所能达到的极限有关。只要P/NP问题还是一个未解之谜，人类就无法确切地知道自己所能取得的成就的极限在哪里。这不由得让人感到精神一振。

章节注释和文献

本书的内容基于我对计算复杂度领域的研究经验，以及与几千个来自学术界和工业界同样对P/NP问题感兴趣的研究者。书中的一些内容改编自我的博客“Computational Complexity”（计算复杂度）。

在著书过程中，我从许多资料引用了一些故事、例子和结果。下面将列出所有的资料。

所有对于这些资料或链接的更新，以及文中发现的重大错误，都将公布在本书的网站http://press.princeton.edu/titles/9937.html上。网站也提供了引用的图书、访谈、附加信息以及P/NP问题的拓展阅读等内容。

前言

Lance Fortnow, “The Status of the P versus NP Problem,” *Communications of the ACM* 52, no. 9 (September 2009): 78–86.

Stephen Hawking, *A Brief History of Time: From the Big Bang to Black Holes* (New York: Bantam Dell, 1988).

第**1**章

维露卡·索尔特的故事是从罗尔德·达尔的小说*Charlie and the Chocolate Factory* (New York: Knopf, 1964)中摘录的。

对松冈容子工作的讨论，即她的科研小组研发的解剖学上正确无误的机械手，信息来自2010 CRA Snowbird Conference（2010年7月18日）上的一场讲座。

旅行推销员问题是用Mark Daskin的软件生成的，网址是http://sitemaker.umich.edu/msdaskin/software。

第2章

除了2.4节，本章其余内容都是我虚构的，目的是为了描绘不太可能发生的P=NP的世界。

第3章

关于米尔格拉姆的实验，参见Stanley Milgram, “The Small World Problem,” *Psychology Today* 2, no. 1 (1967): 60–67。

贝肯分数的计算来自Internet Movie Database。

对于四色问题有一本可读性更强的故事，参见Robin Wilson, *Four Colors Suffice: How the Map Problem Was Solved* (Princeton, NJ: Princeton University Press, 2004)。

第4章

引用库克的话其实有修改，经过了一些现代术语的替换。原文如下：

> The theorems suggest that {tautologies} is a good candidate for an interesting set not in L*, and I feel it is worth spending considerable effort trying to prove this conjecture. Such a proof would be a major breakthrough in complexity theory.

Steve Cook, “The Complexity of Theorem-Proving Procedures,” in *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (New York: ACM), 151–158.

卡普的后续论文指的是：Richard Karp, “Reducibility among Combinatorial Problems,” *Complexity of Computer Computations* 40, no. 4 (1972): 85–103。

Bob Sehlinger (author) and Len Testa (contributor), *The Unofficial Guide*

Walt Disney World 2010 (New York: Wiley, 2010).

4.3节内容取材自Donald Knuth, “A Terminological Proposal,” *ACM SIGACT News* 6, no. 1 (January 1974): 12–18。

Kevin Sack, “60 Lives, 30 Kidneys, All Linked,” *New York Times*, February 19, 2012, A1.

第**5**章

本章从以下资料汲取了很多素材。

Lance Fortnow and Steve Homer, “A Short History of Computational Complexity,” *Bulletin of the European Association for Theoretical Computer Science* 80 (June 2003)，“Computational Complexity” 专栏，以及与许多研究者的个人讨论（包括斯蒂芬·库克和列昂尼德·莱文）。

Dennis Shasha and Cathy Lazere, “A Good Solution Is Hard to Find,” in *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists* (New York: Springer, 1995).

Juris Hartmanis, “Observations about the Development of Theoretical Computer Science,” *Annals of the History of Computing* 3, no. 1 (January 1981): 42–51.

B. A. Trakhtenbrot, “A Survey of Russian Approaches to Perebor (Brute-Force Search) Algorithms,” *Annals of the History of Computing* 6, no. 4 (October 1984): 384–400.

Michael Sipser, “The History and Status of the P versus NP Question,” in *Proceedings of the 24th Annual ACM Symposium on Theory of Computing* (New York: ACM, 1992), 603–618. 这篇文章中包含了哥德尔写给冯·诺依曼的信件副本和英文译文。

柯尔莫哥洛夫曾经试图成为历史学家的故事得到了几个俄罗斯人的印证，时间是在2003年4月27日至5月2日，地点是在德国的Dagstuhl召开的

柯尔莫哥洛夫复杂度及应用百年座谈会上。这个故事也记录在我的博客“Computational Complexity”上，时间是2003年5月1日。

参考文献

Alan Cobham, “The Intrinsic Computational Difficulty of Functions,” in *Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Science*, 24–30.

Stephen Cook, “The Complexity of Theorem-Proving Procedures,” in *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (New York: ACM, 1971), 151–158.

Jack Edmonds, “Paths, Trees and Flowers,” *Canadian Journal of Mathematics* 17 (1965): 449–467.

Juris Hartmanis and Richard Stearns, “On the Computational Complexity of Algo- rithms,” *Transactions of the American Mathematical Society* 117 (1965): 385–406.

Richard Karp, “Reducibility among Combinatorial Problems,” *Complexity of Computer Computations* 40, no. 4 (1972): 85–103.

Leonid Levin, “Universal Sequential Search Problems” [in Russian], *Problemy Pred. Informatsii* 9, no. 3 (1971): 265–266. Translation in B. A. Trakhtenbrot, “A Survey of Russian Approaches to Prebor (Brute-Force Search) Algorithms,” *Annals of the History of Computing* 6, no. 4 (October 1984): 384–400.

Warren McCulloch and Walter Pitts, “A Logical Calculus of the Ideas Immanent in Nervous Activity,” *Bulletin of Mathematical Biology* 5, no. 4 (1943): 115–133.

Panel discussion, *Complexity of Computer Computations* 40, no. 4 (1972): 169–185.

Alan Turing, “On Computable Numbers, with an Application to the

Entscheidungsproblem,” *Proceedings of the London Mathematical Society* 42 (1936): 230–265.

S. Yablonsky, “On the Impossibility of Eliminating PEREBOR in Solving Some Problems of Circuit Theory,” *Doklady Akademii Nauk SSSR* 124 (1959): 44–47.

Y. Zhuravlev, “On the Impossibility of Constructing Minimal Disjunctive Normal Forms for Boolean Functions by Algorithms of a Certain Class,” *Doklady Akademii Nauk SSSR* 132 (1960): 504–506.

第**6**章

旅行推销员问题[1]的例子来自“CRPC Researchers Solve Traveling Salesman Problem for Record-Breaking 13,509 Cities”，2003年由莱斯大学并行计算研究中心举行的一次新闻发布会。

1 旅行推销员问题是用Mark Daskin 的软件生成的，网址是http://sitemaker.umich.edu/msdaskin/software。

为了找到地图填色的启发式方法的例子，我向网络社区寻求了帮助，在答疑网站（http://cstheory.stackexchange.com/questions/4027/coloring-planar-graphs）和我的博客上发布了帖子。

敌友国各省地图取材于于David P. Dailey, “Uniqueness of Colorability and Colorability of Planar 4-Regular Graphs Are NP-Complete”, *Discrete Mathematics* 30 (1980): 289–293。

第**7**章

第一句中引用尤里斯·哈特马尼斯的话来自于他1985年春季在康奈尔大学讲的一堂课上。

关于P/NP策略，参见JACM的相关页面（http://jacm.acm.org/instructions/pnp）。

维纳里·德奥拉利卡的P/NP论文附加在他于2010年8月6日发送的一封电子邮件中，收信人是我和其他21个研究者。

吉罗拉莫的故事参见David Kahn, *The Codebreakers: The Story of Secret Writing* (New York: Macmillan, 1967)。

第**8**章

本章关于密码学早期历史的叙述很多都来自David Kahn的著述，*The Codebreakers: The Story of Secret Writing* (New York: Macmillan, 1967)。

零知识证明数独的例子源于我在2006年8月3日撰写的一篇博文，发布在我的博客“Computational Complexity”上：http://blog.computationalcomplexity.org/2006/08/zero-knowledge-sudoku.html。

参考文献

Whitfield Diffie and Martin Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory* 22, no. 6 (November 1976): 644–654.

Craig Gentry, “Fully Homomorphic Encryption Using Ideal Lattices,” in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing* (New York: ACM, 1979), 169–178.

Ronald Rivest, Adi Shamir, and Leonard Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” *Communications of the ACM* 21, no. 2 (February 1978): 120–126.

第**9**章

我对理查德·费曼在量子计算学中起到的作用的叙述取材自David Deutsch的著作，“Quantum Computation,” *Physics World*, January 6,

1992。

参考文献

Charles Bennett and Gilles Brassard, “Quantum Cryptography: Public Key Distribution and Coin Tossing,” *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing* (Amsterdam: Elsevier, 1984), 175–179.

Charles Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters, “Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels,” *Physical Review Letters* 70 (1993): 1895–1899.

Lov Grover, “A Fast Quantum Mechanical Algorithm for Database Search,” in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (New York: ACM, 1996), 212–219.

Stephen Pincock, *Codebreaker: The History of Codes and Ciphers* (New York: Walker and Co., 2006), 151, for the Bennett and Brassard beach story.

Peter Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Loga- rithms on a Quantum Computer,” *SIAM Journal on Computing* 26 (1997): 1484–1509.

第**10**章

摩尔定律来自Gordon Moore，“Cramming More Components onto Integrated Circuits,” *Electronics* 38, no. 8 (April 19, 1965)。

华生的硬件配置公布在IBM的“What Runs IBM Watson and Why”博文中， 作者是David Davidian。

集装箱的故事来自Marc Levinson, *The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger* (Princeton, NJ: Princeton University Press, 2008)。

大数据的统计信息的来源

http://www.youtube.com/t/press_statistics

http://techcrunch.com/2011/03/14/new-twitter-stats-140m-tweets-sent-per-day-460k-accounts-created-per-day/

http://www.facebook.com/press/info.php?statistics

http://email.about.com/od/emailtrivia/f/emails_per_day.htm

http://public.web.cern.ch/public/en/lhc/Computing-en.html

http://space.about.com/od/telescopesandoptics/p/hubbleinfo.htm

http://webbtelescope.org/webb_telescope/technology_at_the_extremes/quick_

http://royal.pingdom.com/2011/01/12/internet-2010-in-numbers/

人名表

A

Adi Shamir 阿迪·沙米尔
Alan Cobham 艾伦·科巴姆
Alan Turing 阿兰·图灵
Albert Einstein 阿尔伯特·爱因斯坦
Albert Meyer 艾伯特·梅耶
Alexander Razborov 亚历山大·拉兹波洛夫
Alexey Liapunov 阿列克谢·李雅普诺夫
Alfred Kempe 阿尔弗雷德·肯普
Alonzo Church 阿隆索·丘奇
Andrew Wiles 安德鲁·怀尔斯
Andrey Kolmogorov 安德烈·柯尔莫哥洛夫
Arthur Conan Doyle 亚瑟·柯南·道尔
Arthur Scherbius 亚瑟·谢尔比乌斯

B

Boris Trakhtenbrot 鲍里斯·特拉腾布罗
Brian Hammer 布莱恩·哈默

C

Charlie Bennett 查理·贝内特
Charlie Chaplin 查理·卓别林
Claude Berge 克劳德·伯杰
Claude Shannon 克劳德·香农
Clifford Cocks 克利福德·柯克斯
Craig Gentry 克雷格·金特里
Cy Young 赛·杨

D

David Hilbert 大卫·希尔伯特
David Williamson 大卫·威廉森

E

Edgar Allan Poe 埃德加·爱伦·坡
Edward Kasner 爱德华·卡斯纳
Eli Wallach 伊莱·瓦拉赫

F

Francis Guthrie 弗朗西斯·格思里

G

George Brown 乔治·布朗
Georg Dantzig 格奥尔格·丹齐克
Gerolamo Cardano 吉罗拉莫·卡尔达诺
Gilles Brassard 吉尔斯·布拉萨德
Gordon Moore 戈登·摩尔
Gregory Chaitin 格里高利·蔡廷
Grigori Perelman 格里高利·佩雷尔曼

H

Humphrey Bogart 亨弗莱·鲍嘉

I

Isaac Newton 艾萨克·牛顿

J

Jack Edmonds 杰克·埃德蒙兹
James Webb 詹姆斯·韦伯
Joe DiMaggio 乔·狄马乔
Joe Mitchell 乔·米切尔
Johannes Kepler 约翰内斯·开普勒
John Nash 约翰·纳什
John von Neumann 约翰·冯·诺依曼
Julia Roberts 茱莉娅·罗伯茨
Julius Caesar 朱利叶斯·凯撒
Juris Hartmanis尤里斯·哈特马尼斯

K

Kenneth Appel 肯尼斯·阿佩尔
Ketan Mulmuley 凯坦·马尔马利
Kevin Bacon 凯文·贝肯
Knuth 高德纳
Kurt Gödel 库尔特·哥德尔

L

Lance Fortnow 兰斯·福特诺
Leonard Adleman 伦纳德·阿德曼
Leon Batista Alberti 莱昂·巴蒂斯塔·阿尔伯蒂
Leonhard Euler 莱昂哈德·欧拉
Leonid Khachiyan 利奥尼德·卡奇安
Leonid Levin 列昂尼德·莱文
Lov Grover 洛夫·格罗佛

M

Malcolm McLean 马尔科姆·麦克莱恩
Manindra Agrawal 曼尼达·阿加拉瓦尔
Manuel Blum 曼纽尔·布卢姆
Martin Hellman 马丁·赫尔曼
Michael Sipser 迈克尔·塞普斯
Michel Goemans 米歇尔·戈麦斯
Milena Pavel 米莱娜·帕维尔
Minsk Borov 明斯克·波洛夫
Moshe Vardi 莫舍·瓦迪
Muhammad ibn Mūsā al-Khwārizmī 穆罕默德·伊本·穆萨·花剌子密

N

Narendra Karmarkar 纳伦德拉·卡马卡
Neeraj Kayal 尼拉吉·卡亚尔
Nicholas Pippenger 尼古拉斯·皮彭格
Nitin Saxena 尼汀·萨克斯纳
Noam Chomsky 诺姆·乔姆斯基

P

Paul Erdős 保罗·爱多士
Pete Johnson 皮特·约翰逊
Peter Shor 彼得·肖尔
Peter Tait 彼得·泰特
Pierre de Fermat 皮埃尔·德·费马
Pyotr Novikov 彼得·诺维科夫

R

Ray Solomonoff 雷·索洛莫诺夫

René Descartes 勒内·笛卡尔
Richard Feynman 理查德·费曼
Richard Karp 理查德·卡普
Richard Stearns 理查德·斯特恩斯
Roald Dahl 罗尔德·达尔
Roger Merkle 罗杰·默克尔
Ronald Rivest 罗纳德·李维斯特

S

Sanjeev Arora 桑吉弗·阿罗拉
Sergey Yablonsky 谢尔盖·雅布隆斯基
Sherlock Holmes 歇洛克·福尔摩斯
Sophia Loren 索菲亚·罗兰
Stephen Hawking 史蒂芬·霍金
Stephen Kleene 斯蒂芬·克莱尼
Steve Cook 斯蒂芬·库克
Steve Frank 史蒂夫·弗兰克
Steven Rudich 史蒂文·鲁迪赫
Subhash Khot 萨布哈什·霍特

T

Tim Burton 蒂姆·伯顿
Tom Hull 汤姆·赫尔
Tom Jones 汤姆·琼斯

V

Veruca Salt 维鲁卡·索尔特
Vinary Deolalikar 维纳里·德奥拉利卡

W

Walter Pitts 沃尔特·皮茨
Warren McCulloch 沃伦·麦卡洛克
Whitfield Diffie 惠特菲尔德·迪菲
William of Ockham 奥卡姆的威廉
William Rowan Hamilton 威廉·罗恩·哈密顿
Willie Wonka 威利·旺卡
Winston Churchill 温斯顿·丘吉尔
Wolfgang Haken 沃尔夫冈·哈肯

Y

Y. M. Barzdin Y. M. 巴兹丁
Yoky Matsuoka 松冈容子

Z

Zhuravlev 佐拉夫勒夫

05. 轻松撰写功能规格书 - 第一部分： 为什么要写？

2000年10月2日，星期一

乔尔测试面世之初， 读者反映其中最难做到的部分就是写规格书。 写软件规格书就好像像漱口： 大家都觉得是件好事， 但就是没人做。

为什么人们不写规格书？ 有人说跳过写规格书的阶段可以节省时间。 这些人眼中写规格书是一种奢侈， 只有那些为NASA制造太空飞船的工程师， 或是大型知名保险公司的工作人员们才配享有。 胡说八道。 首先， 不写规格书是软件项目开发中最大的不必要风险， 其愚蠢程度就像不带装备就横穿莫哈维沙漠，心存侥幸能逃过一劫。 那些不写规格书就一头扎进代码中的程序员和软件工程师， 以为自己是耍枪高手， 不用瞄准就能拔枪射击。 他们在做梦。 他们的效率非常低。 他们写的代码烂、做出的软件更烂， 并且为项目平添了非常大的、完全不必要的风险。


我认为对于任何非玩票性质的项目（开发时间超过一周或人员多于一名）， 不写规格书会导致项目花费的时间更长、代码的质量更差。 原因如下。

规格书最重要的功能就是对程序进行设计。 即使只有你自己在写代码， 规格书也只写给自己看， 在写规格书的过程中， 通过详细描述程序的工作原理， 你会被迫对程序做真正的设计。

让我们虚构来自两家公司的两个程序员。 来自急速香蕉软件的史比迪小姐，从不写规格书。“规格书？我们才不需要那玩意！” 来自好脾气软件公司的罗杰斯先生， 在写代码之前总会完成详细的规格书。 我有好几位假想中的朋友，这两位也在其中。

史比迪和罗杰斯有一个共同点： 他们都负责维护各自产品2.0版本的历史兼容性。

史比迪决定提供历史兼容性的最好方法是写一个简单的转换器， 把1.0版本文件转换为2.0版本。 她立刻开始敲代码。 敲，敲，敲。点，点，点。 硬盘飞转， 尘土飞扬。 两周后， 她的转换器完工了。 但史比迪的客户不满意，因为她的代码强迫公司的所有人都必须立即升级到新版本。 史比迪最大的客户 Nanner Splits Unlimited公司拒绝购买新的软件。

Nanner Splits需要2.0版本不用转换就能处理1.0版本的文件。 史比迪决定写一个降级转换器， 并把它 绑到“保存”按钮上。 这下乱套了， 因为虽然看上去你可以使用2.0版本的功能， 但在保存时，由于只支持1.0版本的格式， 软件会提示你半个小时前用过的功能在旧文件格式上不可用。 写降级转换器不仅又花了两周时间， 而且效果还不好。 这样总共花费 了4周时间。

再来看好脾气软件公司的罗杰斯先生，他古怪偏执的一面表现在没有规格书就拒绝写代码。 他花了20分钟来做历史兼容功能的设计，想法和史皮迪差不多， 规格书大致的意思是：

- 打开一个用旧版本产品创建的文件， 自动将其转换成新版本的格式。

这个规格书拿给客户一看， 客户急了：“等一下！ 我们不想强迫每个人立刻升级！” 于是罗杰斯先生又想了想， 修改了说明书，如下：

- 打开一个用旧版本产品创建的文件， 会在内存中自动将其转换成新版本的格式。 保存文件的时候， 用户可以选择把文件转换回旧版本。

又过了20分钟。

罗杰斯先生的上司是一个面向对象编程的狂人， 他看到规格书，觉得缺点什么。 他给出了一个不同的架构。

- 重构代码， 使用两个接口：V1和V2。 V1包含所有版本一的功能，V2继承自V1， 增加所有的新功能。 V1::Save负责处理历史兼容性， V2::Save用新格式来保存文件。 如果你打开V1文件时想使用V2的功能， 软件会发出提醒， 用户要么转换文件， 要么放弃使用

新功能。

20分钟过去。

罗杰斯先生郁闷了。 重构将耗时3个星期， 而不是他预计的2个星期！ 但是这么做确实优雅地解决了用户的所有问题， 所以他接受了方案， 开始编程。

罗杰斯先生总共用了3周零1小时。 史皮迪总共用了4周， 但代码的质量不那么好。

这个故事的教训是， 瞎编一个例子你可以证明任何事情。 额， 我不是这个意思。 故事的教训是， 当你用人类的语言设计产品时， 可以在几分钟的时间里权衡几种方案的利弊，然后修正、改进你的设计。 谁也不会为了在文字处理器里删除一段话而难过。 但如果你选择编程的语言里做产品的设计， 可能要花 数周的时间才能做出可以让人反馈的设计。 更糟的是， 程序员对于耗时两周时间写出的代码是有感情的， 不管这些代码错得多离谱。 无论史皮迪的上司或是客户说什么， 她都不会愿意丢掉她辛苦写出的、漂亮的转换器代码， 尽管这段代码在架构上完全谈不上漂亮。 结果是， 最后交付的产品形态介于最初错误的设计以及理想的设计之间。 人们会评论它“是我们能做的最好的设计了，毕竟我们辛辛苦苦写了那么多代码， 总不能全都推倒重来吧。” 这个评价远比不上：“这是我们能做的最好的设计，没有之一。”

以上是写规格书的第一大理由。 第二大 理由是节省沟通的时间。 通过写规格书，你只需要和别人沟通一次， 团队的其他成员只要去读规格书， 就能知道程序的很多方面。 测试人员通过规格书，可以了解程序有哪些符合预期的功能和行为， 也就是该从哪些角度去测试。 营销人员以规格书为蓝本，撰写出充满溢美之词的产品白皮书，然后放到网站上去宣传你们还未出世的杰作。 业务拓展人员经常误读规格书， 然后用天马行空般的想象力把产品 塑造成秃顶或是皮肤疣（warts）的克星， 不过只要他们能忽悠到投资，你就没什么可抱怨的。 开发者读了规格书，就知道 该写哪些代码。 客户读了规格书， 就能确信开发者们正在做的产品是物有所值的。 技术文档编写者以规格书为参考， 写出给客户看的图文并茂的说明书（就是那些经常遗失或被丢弃的小册

子， 不过那是另一个故事了）。经理们读了规格书， 然后就可以在管理会议上假装一切尽在他们掌控之中。 等等。

如果你没写规格书，所有这些沟通的过程还是要发生， 因为它们总会发生， 不过这种情况下，它们会一次又一次地发生。 测试人员拿到软件只能随便点点看看， 因为他们不清楚软件怎么样才算正常工作， 一旦发现有奇怪的程序行为， 他们就会一再地问程序员各种愚蠢的问题。 这不仅会导致程序员的工作效率被破坏殆尽 ,而且程序员在回答这些问题时 倾向于用代码实现细节来解释， 而给不出所谓“恰当的答案”。 也就是说， 测试人员是基于程序来测试程序， 而不是基于程序来测试设计， 而后者可能，嗯，还稍微更有用一点。

如果不写规格书， 发生在可怜的技术文档编写者身上的事才叫滑稽（带有一丝悲凉的感觉）。他们通常 比较弱势， 不敢打扰程序员大人们。 许多公司的情况是， 如果技术文档编写者养成了向程序员提问的习惯， 比如某个软件 怎么样才算正常工作， 程序员就会向经理告状，说自己被那些[形容词已被和谐]的文档编写者烦得什么都干不成， 能不能 别让他们靠近我。 经理们为了提高工作效率， 只能禁止文档编写者再去浪费程序员们的宝贵时间。 至于这种情况发生在哪些公司， 你从他们的帮助文件和产品手册上就看得出来， 里面包含的信息比运行软件时你自己得到的信息还少。 当你看到屏幕上出现如下提示信息

- 你想开启LRF-1914支持功能吗？

然后你点“帮助”， 弹出的帮助主题会让你哭笑不得：

- 让你在使用LRF-1914支持功能（默认）和不使用LRF-1914支持功能之间切换。 如果你想使用LRF-1914支持功能， 选择“是”或按“Y”。 如果你不想使用LRF-1914支持 功能， 选择“否”或按“N”键。

额，谢谢了。 很明显文档编写者在努力掩盖自己根本不知道*LRF-1914*支持功能是什么的事实。 他们不能问程序员，可能因为（a）他们不好意思开口，或者（b）程序员人品很差而且远在伦敦，或者（c） 管理层禁止他们打扰程序员， 或是因为其他的难以列举的大公司通病。 但是最根本的问题在于， 没有规格书可以做参考。

写规格书的第三大理由是没有详细的说明， 就无法制定计划表。 如果你在攻读博士学位而且做好了为之奋斗14年的准备，你可以没有计划表。 或者如果你写的程序有望成为下一个“毁灭公爵”* 一个划时代的3D游戏——译者注，那我们可以理直气壮地说“什么时候把程序做结实做漂亮了，我们就什么时候交付”， 你也可以没有计划表。 但几乎所有面向实际业务的开发工作， 你必须要知道要做某件事需要多长的时间， 因为开发产品是要花钱的。 买牛仔裤之前还要看价格呢， 何况是自负盈亏的投资者，他们在决定是否开发某个产品时， 当然需要先知道开发的时间成本，进而换算成经济成本。 关于如何做计划，请阅读我写的轻松掌控软件开发进度。

公司常犯的一个重大错误是， 当大家对某些设计有争议时，争议从来都得不到解决。 Windows 2000 的开发主管布莱恩·瓦伦丁有一句名言: “如果10分钟以内无法达成决议， 那就一直商量到达成决议为止。”

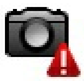
太多的软件开发机构都有这个问题： 一旦出现设计上的争议， 通常出于某种政治原因， 没有人站出来力挽狂澜，促成一个决议。 这导致程序员们只能就无关痛痒的设计进行开发。 随着时间的推移， 所有困难的决定 都被推迟到了最后一刻。 这样的项目是最有可能失败的项目。 如果你开了一家研发某项新技术的初创公司， 并且发现 公司经常性地无法做出决定， 那你还不如趁早关门， 把钱还给投资者， 因为这样下去公司什么也做不成。

在写规格说明书的过程中，所有那些令人不悦的或大或小的设计决定都会统统暴露出来，如果不写，它们很容易就被掩盖过去了。 即使是非常琐碎的决定， 也能从规格书中看出问题的端倪。 比如你们在做一个会员制的网站， 可能大家都同意如果会员忘了密码， 就通过邮件告知他们。 很好。 但是这还不足以指导开发。 要写出代码， 你需要知道那封电子邮件的确切措词。 大部分公司都不信任程序员的文字水平，不让他们直接撰写真正要展示给客户看的文案（而这种不信任通常是有道理的）。所以消息的具体措词，需要交给某个营销、公关或是文学专业的人去拟定。

“亲爱的客户， 这是你忘记的密码。 下次别这么大意了。” 当你强迫自己把规格书写得规范、完整的时候（我马上就会讲这个）， 你就会注意到很多细节的问题， 要么你可以立即解决他们， 要么就在旁边做一

个大大的红色记号。

好，现在大家意见一致了吧。 规格书必不可少，就好像每个人都有母亲，每个人都爱苹果派。 我怀疑大部分人都明白这一点， 而我发了 这么多牢骚， 只是贻笑大方， 并没有什么教育意义。 问题是为什么人们不写规格书呢？ 肯定不是为了节省时间， 实际上恰恰相反， 而且我觉得大多数程序员都对此深有体会。（在大部分机构里 唯一能算作“规格书”的东西是那种生硬的，篇幅只有一页的文本文档，还是程序员在向300个人解释过那些该死的功能之后，愤愤然一挥而就的。）

 我认为是因为有太多的人不喜欢写作了。 面对一张空白的屏幕，很多人都会发怵。 我个人克服了这种恐惧， 得益于我在大学里选过一门课， 课程要求每周交一篇3至5页的小品文。 写作能力也是一种肌肉。 你写得越多，就越能写。 如果你需要写规格书但写不出来， 可以先开始记日记，写博客，参加一个创作培训班， 或者干脆给认识的人写信，不管是亲戚也好， 你4年多没联系的大学室友也罢。 不要放过任何一个把词写在纸上的机会，持续改进你的规格书写作技巧。 如果你是软件开发活动的管理者， 要是你手下的人不愿意写规格书， 就把他们送去参加一个为期两周的创作培训班， 最好是在深山里举办的那种。

如果你没有在一个要求写功能规格书的公司工作过， 你可能不知道这种文档时什么样的。 在这个系列的下一篇文章里， 我会示范一个简短的规格书，供你参考， 然后再看看一份好的规格书应该包括什么。 请继续阅读！

Table of Contents